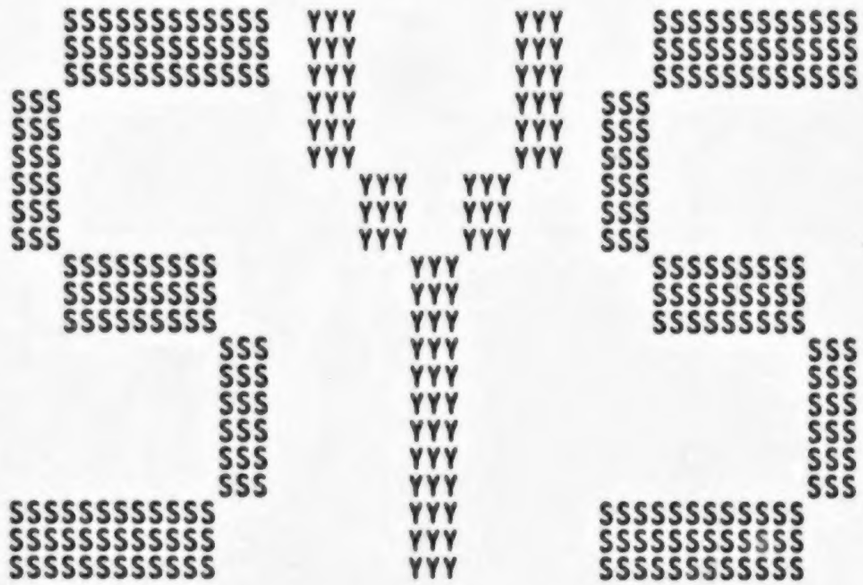
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

FILEID**SYSGETJPI

```
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT          JJ  PPPPPPPP     IIIIII
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT          JJ  PPPPPPPP     IIIIII
SS          YY      YY SS          GG          EE              TT              JJ  PP      PP     II
SS          YY      YY SS          GG          EE              TT              JJ  PP      PP     II
SS            YY  YY   SS          GG          EE              TT              JJ  PP      PP     II
SS            YY  YY   SS          GG          EE              TT              JJ  PP      PP     II
  SSSSSS        YY       SSSSSS    GG          EEEEEEEE        TT              JJ  PPPPPPPP       II
  SSSSSS        YY       SSSSSS    GG          EEEEEEEE        TT              JJ  PPPPPPPP       II
        SS      YY              SS  GG  GGGGGG  EE              TT          JJ  JJ  PP             II
        SS      YY              SS  GG  GGGGGG  EE              TT          JJ  JJ  PP             II
        SS      YY              SS  GG      GG  EE              TT          JJ  JJ  PP             II          ....
        SS      YY              SS  GG      GG  EE              TT          JJ  JJ  PP             II          ....
SSSSSSSS        YY      SSSSSSSS    GGGGGG    EEEEEEEEEE      TT            JJJJJJ  PP           IIIIII        ....
SSSSSSSS        YY      SSSSSSSS    GGGGGG    EEEEEEEEEE      TT            JJJJJJ  PP           IIIIII        ....

LL              IIIIII    SSSSSSSS
LL              IIIIII    SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  SYSGETJPI - GET JOB PROCESS INFORMATION SYSTEM SERVICE
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;****************************************************************************
0000     5  ;*                                                                          *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                    *
0000     9  ;*                                                                          *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15  ;*  TRANSFERRED.                                                            *
0000    16  ;*                                                                          *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19  ;*  CORPORATION.                                                            *
0000    20  ;*                                                                          *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23  ;*                                                                          *
0000    24  ;*                                                                          *
0000    25  ;****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY: VMS Executive, System services.
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;
0000    32  ;       Return accounting, quota, and informational data about the current
0000    33  ;       process, or any other process.
0000    34  ;
0000    35  ; ENVIRONMENT: Kernel Mode
0000    36  ;
0000    37  ; AUTHOR: Henry M. Levy , CREATION DATE: 20-October-1977
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-024 MSH0071         Michael S. Harvey       26-Jul-1984
0000    42  ;               Don't clobber user address space when issuing JPI items
0000    43  ;               that get serviced via kernel ASTs.
0000    44  ;
0000    45  ;       V03-023 MSH0062         Michael S. Harvey       6-Jul-1984
0000    46  ;               Don't skip the NULL process when wildcarding through
0000    47  ;               the process set.
0000    48  ;
0000    49  ;       V03-022 MSH0040         Michael S. Harvey       1-May-1984
0000    50  ;               Look for image name in designated area whenever an AME
0000    51  ;               is running in the process.
0000    52  ;
0000    53  ;       V03-021 HWS0055         Harold Schultz          11-Apr-1984
0000    54  ;               Add JPI$_MASTER_PID item so that the PID of the
0000    55  ;               master process in a job can be accessed.
0000    56  ;               (since the MPID in the JIB is in internal format, it
0000    57  ;                is translated to extended format before it is returned)
```

```
0000     58 ;              Change JPI$_PROC_INDEX special processing entry point to
0000     59 ;              be local.
0000     60 ;
0000     61 ;       V03-020 CWH3020         CW Hobbs                20-Mar-1984
0000     62 ;              Add JPI$_PROC_INDEX item so that applications which used to
0000     63 ;              look at the low word of the PID can adapt.
0000     64 ;
0000     65 ;       V03-019 MSH0010         Michael S. Harvey       14-Feb-1984
0000     66 ;              Restructure internal item information tables to
0000     67 ;              accomodate counted strings up to 255 bytes.
0000     68 ;
0000     69 ;       V03-018 KFH0009         Ken Henderson            7 Sep 1983
0000     70 ;              Fix KFH0008 more thoroughly
0000     71 ;              Also add SPC_MODE routine.
0000     72 ;
0000     73 ;       V03-017 KFH0008         Ken Henderson           29 Aug 1983
0000     74 ;              Add documentation on how
0000     75 ;              to add itemcodes.
0000     76 ;
0000     77 ;       V03-016 KFH0008         Ken Henderson           18 Aug 1983
0000     78 ;              Don't return ACCVIO if can't get data
0000     79 ;
0000     80 ;       V03-015 WMC0001         Wayne Cardoza           28-Jul-1983
0000     81 ;              Allow chained item lists.
0000     82 ;
0000     83 ;       V03-014 KFH0007         Ken Henderson           18 Jul 1983
0000     84 ;              Fixed IMAGNAME bug in source.
0000     85 ;
0000     86 ;       V03-013 KFH0006         Ken Henderson           12 Jul 1983
0000     87 ;              Fixed STRDSC bug in source.
0000     88 ;
0000     89 ;       V03-012 KFH0005         Ken Henderson           16 Jun 1983
0000     90 ;              Fixed various bugs in source.
0000     91 ;
0000     92 ;       V03-011 KFH0004         Ken Henderson           27 May 1983
0000     93 ;              Added HEXSTR datatype to macro.
0000     94 ;              Fixed wildcarding bug, DELPEN bug,
0000     95 ;              and sKAST buffer length bug. Cleaned
0000     96 ;              up use of stack scratch space.
0000     97 ;
0000     98 ;       V03-010 KFH0003         Ken Henderson           24 Mar 1983
0000     99 ;              Fetch other processes' PHD items directly
0000    100 ;              if the header is resident, and allow
0000    101 ;              NULL and SWAPPER processes to be visible.
0000    102 ;
0000    103 ;       V03-009 KFH0002         Ken Henderson            1 Mar 1983
0000    104 ;              Mods to support bitfield item-codes.
0000    105 ;
0000    106 ;       V03-008 CWH1002         CW Hobbs                25-Feb-1983
0000    107 ;              Modify to use extended pids.  Use SCH$C_SWPPIX to locate the
0000    108 ;              pcb of the swapper.
0000    109 ;
0000    110 ;       V03-007 KFH0001         Ken Henderson           10 Feb 1983
0000    111 ;              Condense the table macros into JPI_ITEM_CODE,
0000    112 ;              and add call the JPI_GENERATE_TABLE to invoke
0000    113 ;              JPI_ITEM_CODE for each item-code.
0000    114 ;
```

```
0000   115 ;       V03-006 LJK0187         Lawrence J. Kenah       22-Oct-1982
0000   116 ;               Correct erroneous reference to ASTCNT. Make routine that checks
0000   117 ;               accessibility of image name a global routine.
0000   118 ;
0000   119 ;       V03-005 LJK0157         Lawrence J. Kenah       7-Apr-1982
0000   120 ;               Add support for JPI$_IMAGECOUNT for LIB$SPAWN's benefit
0000   121 ;
0000   122 ;       V03-004 LJK0155         Lawrence J. Kenah       1-Apr-1982
0000   123 ;               Handle quota deductions in consistent fashion.
0000   124 ;
0000   125 ;       V03-003 MSH0001         Maryann Hinden          23-Mar-1982
0000   126 ;               Fix broken BSBW's.
0000   127 ;
0000   128 ;       V03-002 DWT0032         David Thiel             22-Mar-1982
0000   129 ;               Correct length of probe for returning asynchronous result
0000   130 ;               length.
0000   131 ;
0000   132 ;       V03-001 LJK0146         Lawrence J. Kenah       16-Mar-1982
0000   133 ;               Correct bugs along code path for not enough nonpaged pool.
0000   134 ;               Use action routine to convert AUTHPRI item. Do not return
0000   135 ;               address items for another process. Add additional check
0000   136 ;               that item list has not changed before second scan.
0000   137 ;
0000   138 ;--
```

```
0000    140 ;                   GUIDE TO GETJPI/GETSYI/GETDVI
0000    141 ;                   -----------------------------
0000    142 ;
0000    143 ;Overview
0000    144 ;--------
0000    145 ;
0000    146 ;These three system services are table-driven. The macro definition files
0000    147 ;that help define their tables are shared with DCL and the RTL. This results
0000    148 ;in new item-codes becoming useable with DCL's F$GETXXI lexical functions and
0000    149 ;the RTL's LIB$GETXXI routines automatically. Additionally, new SYSBOOT
0000    150 ;parameters become item-codes to the GETSYIs.
0000    151 ;
0000    152 ;The macro definition files are called JPITABLE.MAR, SYITABLE.MAR, and
0000    153 ;DVITABLE.MAR, and live in MASD$:<VMSLIB.SRC>. During a systembuild, they
0000    154 ;are inserted into the library SYS$LIBRARY:SYSBLDMLB.MLB. DCL and the RTL
0000    155 ;and SYS use this library to define their GETXXI tables. The system
0000    156 ;parameter file <SYS.SRC>SYSPARAM.MAR has also been conditionalized to be
0000    157 ;used to define GETSYI item-codes and is also inserted into SYSBLDMLB.MLB.
0000    158 ;
0000    159 ;
0000    160 ;NOTE:  SYSBLDMLB.MLB is a general macro library for holding macro
0000    161 ;       definitions that are shared between facilities, but will not
0000    162 ;       ship to the customer.
0000    163 ;
0000    164 ;
0000    165 ;When adding an item-code, at least two files need to be edited. One of the
0000    166 ;macro files listed above, as well as an SDL file that defines the 16-bit
0000    167 ;number which is the user-visible item-code. Also, if a SYSBOOT parameter is
0000    168 ;added, an SDL file needs to be updated to define the new GETSYI item-code.
0000    169 ;
0000    170 ;The GETDVI service actually uses only one table, but the GETSYI and GETJPI
0000    171 ;services use several. The JPITABLE file defines all the tables for GETJPI
0000    172 ;and the SYITABLE file defines all the tables for GETSYI. The different
0000    173 ;tables group the pieces of data according to method of retrieval.
0000    174 ;
0000    175 ;In some cases, the piece of data to be returned by the service requires
0000    176 ;special processing to fetch, calculate, or format it before returning it.
0000    177 ;In these cases, the code of the system service needs to be enhanced.
0000    178 ;If the data returned is a new format for DCL, the lexical function
0000    179 ;module of DCL may need to be enhanced. This is also true for the RTL code.
```

```
0000    181 ;The Macros
0000    182 ;----------
0000    183 ;
0000    184 ;A two-level scheme exists for defining the item tables used by the three
0000    185 ;services and the other facilities. A commonly defined macro (called
0000    186 ;JPI_GENERATE_TABLE, SYI_GENERATE_TABLE, or DVI_GENERATE_TABLE) contains
0000    187 ;multiple calls to a lower-level macro (called JPI_ITEM_CODE, SYI_ITEM_CODE,
0000    188 ;or DVI_ITEM_CODE) which actually defines each element in the table.
0000    189 ;While the _GENERATE_TABLE macros are commonly defined, the _ITEM_CODE macros
0000    190 ;are individually defined according to the needs of facility. (For instance,
0000    191 ;the LEXICON module must store the name of the item as an ASCIC string - in
0000    192 ;order to match it with the string supplied in the F$GETXXI function call;
0000    193 ;the other facilities need not store the item name in text.)
0000    194 ;
0000    195 ;When an item-code must be added, an additional call to the _ITEM_CODE macro
0000    196 ;must be added to the appropriate _GENERATE_TABLE macro. In the case of GETJPI
0000    197 ;and GETDVI, the _GENERATE_TABLE macro is defined in the JPITABLE and DVITABLE
0000    198 ;modules. The SYI_GENERATE_TABLE macro is defined by the SYSPARAM module
0000    199 ;- all the calls to the PARAMETER and PQL macros are 'collected' into the
0000    200 ;SYI_GENERATE_TABLE macro. When used in that mode (when GETSYISW is defined),
0000    201 ;the SYI_ITEMTABLES macro also becomes part of the SYI_GENERATE_TABLE macro.
0000    202 ;SYI_ITEMTABLES is defined in the SYITABLE module and contains all the calls
0000    203 ;to the SYI_ITEM_CODE macro that are Not related to SYSBOOT parameters.
0000    204 ;When GETSYISW is defined in SYSPARAM, the PARAMETER macro does not allocate
0000    205 ;or store memory, but rather passes some of the arguments to it on through via
0000    206 ;a call to SYI_ITEM_CODE. That is how all the calls to PARAMETER become calls
0000    207 ;to SYI_ITEM_CODE.
0000    208 ;
0000    209 ;The following is the situation that exists when the symbol GETSYISW is defined.
0000    210 ;The non-SYSBOOT items are defined by the macro SYI_ITEMTABLES in SYITABLE.MAR.
0000    211 ;The SYSBOOT items are defined by each invokation of the PARAMETER macro in
0000    212 ;SYSPARAM.MAR. Note that each invokation of the PQL macro in SYSPARAM.MAR
0000    213 ;invokes the PARAMETER macro twice. When GETSYISW is defined, the PARAMETER
0000    214 ;macro merely passes its arguments through to a call to the SYI_ITEM_CODE
0000    215 ;macro. The SYI_ITEM_CODE macro is locally defined as needed by the facility.
0000    216 ;
0000    217 ;+-------------------------------------------------------------------------+
0000    218 ;|                          SYI_GENERATE_TABLE                             |
0000    219 ;|   SYI_ITEMTABLES                                                        |
0000    220 ;|                                 PARAMETER                 PQL           |
0000    221 ;|                                                  PARAMETER  PARAMETER   |
0000    222 ;|                                                                         |
0000    223 ;|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|
0000    224 ;+-------------------------------------------------------------------------+
0000    225 ;\                           /  \                                       /
0000    226 ;  \                       /      \                                   /
0000    227 ;    FROM SYITABLE.MAR                FROM SYSPARAM.MAR
0000    228 ;    (NON-SYSBOOT ITEMS)               (SYSBOOT ITEMS)
```

```
                                        I 14
0000   230          .SBTTL  DECLARATIONS
0000   231 ;
0000   232 ; INCLUDE FILES:
0000   233 ;
0000   234
0000   235          $ACBDEF                         ; AST control block parameters
0000   236          $DYNDEF                         ; dynamic memory block types
0000   237          $IFDDEF                         ; image file descriptor block
0000   238          $IPLDEF                         ; interrupt priority levels
0000   239          $JIBDEF                         ; define job information block
0000   240          $JPIDEF                         ; define GETJPI item identifiers
0000   241          $PCBDEF                         ; define process control block
0000   242          $PHDDEF                         ; define process header
0000   243          $PSLDEF                         ; processor state longword
0000   244          $STATEDEF                       ; scheduler state definitions
0000   245          $PRDEF                          ; define processor registers
0000   246          $PRIDEF                         ; define priority increment classes
0000   247          $RSNDEF                         ; define resource wait codes
0000   248          $SSDEF                          ; define status codes
0000   249
0000   250 ;
0000   251 ; MACROS:
0000   252 ;
0000   253
0000   254 ;
0000   255 ; Macros to define entries in the six item information tables.
0000   256 ; There is a table for each data structure from which the user may
0000   257 ; request information, and one table for information returned as an
0000   258 ; address.  Tables are indexed by low byte of item identifier.
0000   259 ; Refer to "OWN STORAGE:" for pictures of the table entries.
0000   260 ;
0000   261
0000   262          .MACRO  JPI_ITEM_CODE   BASE,-          ; for service to use
0000   263                                  NAME,-          ; of the item-code
0000   264                                  SOURCE,-        ; of the data
0000   265                                  DTYPE,-         ; of returned value
0000   266                                  BITPOS,-        ; of 'bitval' field
0000   267                                  BITSIZ,-        ; of 'bitval' field
0000   268                                  OUTLEN,-        ; of returned value
0000   269                                  STRUCT          ; containing the field
0000   270
0000   271          .IF NOT_DEFINED JPI$_'NAME
0000   272          .WARN ; JPI$_'NAME' IS NOT DEFINED IN STARDEFFL.SDL
0000   273          .ENDC
0000   274
0000   275          STEP = 4
0000   276          .IIF IDENTICAL <BASE><PCB>,     STEP    = 5
0000   277          .IIF IDENTICAL <BASE><PHD>,     STEP    = 5
0000   278          .IIF IDENTICAL <BASE><ADR>,     STEP    = 5
0000   279          .IIF IDENTICAL <BASE><CTL>,     STEP    = 7
0000   280          .IIF IDENTICAL <BASE><PCBFLD>,  STEP    = 7
0000   281          .IIF IDENTICAL <BASE><PHDFLD>,  STEP    = 7
0000   282
0000   283          .=BASE'TBL+<<JPI$_'NAME&^XFF>*STEP>
0000   284
0000   285          .IIF IDENTICAL <BASE><PCB>,     .WORD   'STRUCT'$'SOURCE
0000   286          .IIF IDENTICAL <BASE><PHD>,     .WORD   'STRUCT'$'SOURCE
```

```
0000    287          .IIF IDENTICAL <BASE><PCBFLD>,  .WORD   PCB$'SOURCE
0000    288          .IIF IDENTICAL <BASE><PHDFLD>,  .WORD   PHD$'SOURCE
0000    289          .IIF IDENTICAL <BASE><ADR>,     .LONG   SOURCE
0000    290          .IIF IDENTICAL <BASE><CTL>,     .LONG   SOURCE
0000    291
0000    292          .IF IDENTICAL <BASE><PCBFLD>
0000    293          .WORD   <'BITSIZ'-1>@11!PCB$V_'BITPOS'
0000    294          .ENDC
0000    295
0000    296          .IF IDENTICAL <BASE><PHDFLD>
0000    297          .WORD   <'BITSIZ'-1>@11!PHD$V_'BITPOS'
0000    298          .ENDC
0000    299
0000    300          .IF DIFFERENT <BASE><ADR>
0000    301
0000    302          $XX$ = VALUE
0000    303          .IIF IDENTICAL <DTYPE><HEXNUM>, $XX$ = VALUE
0000    304          .IIF IDENTICAL <DTYPE><DECNUM>, $XX$ = VALUE
0000    305          .IIF IDENTICAL <DTYPE><HEXSTR>, $XX$ = BSTRING
0000    306          .IIF IDENTICAL <DTYPE><PRVMSK>, $XX$ = BSTRING
0000    307          .IIF IDENTICAL <DTYPE><PRTMSK>, $XX$ = BSTRING
0000    308          .IIF IDENTICAL <DTYPE><PADSTR>, $XX$ = BSTRING
0000    309          .IIF IDENTICAL <DTYPE><CNTSTR>, $XX$ = CSTRING
0000    310          .IIF IDENTICAL <DTYPE><STRDSC>, $XX$ = DSTRING
0000    311          .IIF IDENTICAL <DTYPE><BITVEC>, $XX$ = VALUE
0000    312          .IIF IDENTICAL <DTYPE><BITVAL>, $XX$ = BOOLE
0000    313          .IIF IDENTICAL <DTYPE><STDUIC>, $XX$ = VALUE
0000    314          .IIF IDENTICAL <DTYPE><STDTIM>, $XX$ = BSTRING
0000    315          .IIF IDENTICAL <DTYPE><ACPTYP>, $XX$ = BSTRING
0000    316
0000    317          .BYTE   $XX$
0000    318          .BYTE   OUTLEN
0000    319
0000    320          .ENDC   ; IF DIFFERENT <BASE><ADR>
0000    321
0000    322          .BYTE   JPI$C_'STRUCT'TYPE
0000    323
0000    324          .ENDM   JPI_ITEM_CODE
0000    325
0000    326
0000    327 ;
0000    328 ; This macro defines the entries to the table of special items.
0000    329 ; The items in this table must be handled by action routines
0000    330 ; before being returned.  Each entry has a word item identifier
0000    331 ; followed by the address of an action routine.
0000    332 ;
0000    333
0000    334          .MACRO  SPECIAL_ITEM NAME,ROUTINE
0000    335          .WORD   JPI$_'NAME
0000    336          .ADDRESS ROUTINE
0000    337          .ENDM   SPECIAL_ITEM
0000    338
0000    339 ;
0000    340 ; This macro defines flag bits.
0000    341 ;
0000    342
0000    343          .MACRO  JPIBITS NAME,SIZE
```

```
                     0000   344          JPI_V_'NAME' = JPI_BIT
                     0000   345          JPI_S_'NAME' = SIZE
                     0000   346          JPI_BIT = JPI_BIT + SIZE
                     0000   347          .ENDM   JPIBITS
                     0000   348
                     0000   349 ;
                     0000   350 ; EQUATED SYMBOLS:
                     0000   351 ;
                     0000   352
00000004             0000   353          EFN = 4                             ; event flag number argument
00000008             0000   354          PIDADR = 8                          ; address of PID
0000000C             0000   355          PRCNAM = 12                         ; address of name descriptor
00000010             0000   356          ITMLST = 16                         ; address of item identifiers
00000014             0000   357          IOSB = 20                           ; I/O status block address
00000018             0000   358          ASTADR = 24                         ; ast routine address
0000001C             0000   359          ASTPRM = 28                         ; ast parameter
                     0000   360
                     0000   361 ;
                     0000   362 ; Space is left on stack for routines which may
                     0000   363 ; manipulate values before returning them.
                     0000   364 ;
                     0000   365
FFFFFFD8             0000   366          LOCAL_SPACE = -40
FFFFFFD8             0000   367          SCRATCH = LOCAL_SPACE+0
FFFFFFE8             0000   368          BITDEF = LOCAL_SPACE+16
FFFFFFEC             0000   369          PHDTEMP = LOCAL_SPACE+20
FFFFFFF4             0000   370          WSLIST = LOCAL_SPACE+28
FFFFFFF8             0000   371          DIRCNT = LOCAL_SPACE+32
FFFFFFFC             0000   372          FLAGS = LOCAL_SPACE+36
                     0000   373
                     0000   374 ;
                     0000   375 ; Bit definitions for flags longword on stack
                     0000   376 ;
                     0000   377
00000000             0000   378          JPI_BIT = 0
                     0000   379          JPIBITS WILD,1                      ; we're doing a wildcard operation
                     0000   380          JPIBITS NULLSWAP,1                  ; the target is NULL or SWAPPER
                     0000   381          JPIBITS STRDSC,1                    ; the datatype is a string descriptor
                     0000   382
                     0000   383 ;
                     0000   384 ; Max structure number definitions
                     0000   385 ;
                     0000   386
00000001             0000   387          MAX_ADR_ITEM = <JPI$_LASTADR&^XFF>-1 ; maximum ADRTBL item number
00000010             0000   388          MAX_CTL_ITEM = <JPI$_LASTCTL&^XFF>-1 ; maximum CTLTBL item number
00000025             0000   389          MAX_PCB_ITEM = <JPI$_LASTPCB&^XFF>-1 ; maximum PCBTBL item number
0000001B             0000   390          MAX_PHD_ITEM = <JPI$_LASTPHD&^XFF>-1 ; maximum PHDTBL item number
FFFFFFFF             0000   391          MAX_PCBFLD_ITEM = <JPI$_LASTPCBFLD&^XFF>-1 ; max PCBFLDTBL item number
FFFFFFFF             0000   392          MAX_PHDFLD_ITEM = <JPI$_LASTPHDFLD&^XFF>-1 ; max PHDFLDTBL item number
                     0000   393
                     0000   394 ;
                     0000   395 ; Data type codes (all numeric types have same code)
                     0000   396 ;
                     0000   397
00000000             0000   398          VALUE = 0                           ; numeric value
00000001             0000   399          BSTRING = 1                         ; blank filled string
00000002             0000   400          CSTRING = 2                         ; counted ascii string
```

```
00000003  0000   401          BOOLE = 3                       ; bit value
00000004  0000   402          DSTRING = 4                     ; string descriptor
          0000   403
          0000   404 ;
          0000   405 ; AST control block extensions
          0000   406 ;
          0000   407          $DEFINI ACB
          0000   408
0000001C  0000   409          .=ACB$L_KAST+4                          ;
          001C   410
          001C   411 $DEF     ACB_L_DADDR     .BLKL   1               ; data buffer address
          0020   412 $DEF     ACB_L_EFN       .BLKL   1               ; event flag number
          0024   413 $DEF     ACB_L_IOSB      .BLKL   1               ; completion AST routine addr
          0028   414 $DEF     ACB_L_OPID      .BLKL   1               ; original requester's PID
          002C   415 $DEF     ACB_L_IMGCNT    .BLKL   1               ; PHD$L_IMGCNT of requester
          0030   416 $DEF     ACB_L_COUNT     .BLKL   1               ; item descriptor count
          0034   417 $DEF     ACB_L_ILIST                             ; item descriptor list
          0034   418
0000000C  0034   419          ACB_C_IDESC = 12                        ; item descriptor size
          0034   420
          0034   421          $DEFEND ACB
          0000   422
          0000   423
          0000   424 ;
          0000   425 ; OWN STORAGE:
          0000   426 ;
          0000   427
00000000  0000   428          .PSECT  YF$$SYSGETJPI
          0000   429
          0000   430 ;
          0000   431 ; This array contains the maximum item number for each of the
          0000   432 ; six item data structures, indexed by structure number.
          0000   433 ;
          0000   434
          0000   435 MAXCOUNT:
01        0000   436          .BYTE   MAX_ADR_ITEM
10        0001   437          .BYTE   MAX_CTL_ITEM
25        0002   438          .BYTE   MAX_PCB_ITEM
1B        0003   439          .BYTE   MAX_PHD_ITEM
FF        0004   440          .BYTE   MAX_PCBFLD_ITEM
FF        0005   441          .BYTE   MAX_PHDFLD_ITEM
```

```
                     0006   443 ;
                     0006   444 ; Define the six item data structures.  Each data structure is indexed
                     0006   445 ; by item identifier.
                     0006   446 ;
                     0006   447
                     0006   448 ADRTBL:
                     0006   449                 ;-------------------------;
                     0006   450                 ; .LONG  ADDRESS         ;
                     0006   451                 ; .BYTE  JPI$C_ADRTYPE   ;
                     0006   452                 ;-------------------------;
                     0006   453
00000010             0006   454                 .BLKB   5*<MAX_ADR_ITEM+1>              ; define adr table
                     0010   455
                     0010   456 CTLTBL:
                     0010   457                 ;-------------------------;
                     0010   458                 ; .LONG  ADDRESS         ;
                     0010   459                 ; .BYTE  DTYPE           ;
                     0010   460                 ; .BYTE  LENGTH          ;
                     0010   461                 ; .BYTE  JPI$C_CTLTYPE   ;
                     0010   462                 ;-------------------------;
                     0010   463
00000087             0010   464                 .BLKB   7*<MAX_CTL_ITEM+1>              ; define ctl table
                     0087   465
                     0087   466 PCBTBL:
                     0087   467                 ;-------------------------;
                     0087   468                 ; .WORD  XXX$OFFSET      ;
                     0087   469                 ; .BYTE  DTYPE           ;
                     0087   470                 ; .BYTE  LENGTH          ;
                     0087   471                 ; .BYTE  JPI$C_PCBTYPE   ;
                     0087   472                 ;-------------------------;
                     0087   473
00000145             0087   474                 .BLKB   5*<MAX_PCB_ITEM+1>              ; define pcb table
                     0145   475
                     0145   476 PHDTBL:
                     0145   477                 ;-------------------------;
                     0145   478                 ; .WORD  XXX$OFFSET      ;
                     0145   479                 ; .BYTE  DTYPE           ;
                     0145   480                 ; .BYTE  OUTLEN          ;
                     0145   481                 ; .BYTE  JPI$C_PHDTYPE   ;
                     0145   482                 ;-------------------------;
                     0145   483
000001D1             0145   484                 .BLKB   5*<MAX_PHD_ITEM+1>              ; define phd table
                     01D1   485
                     01D1   486 PCBFLDTBL:
                     01D1   487                 ;-----------------------------;
                     01D1   488                 ; .WORD  XXX$OFFSET         ;
                     01D1   489                 ; .WORD <BITSIZ-1>@11!BITPOS ;
                     01D1   490                 ; .BYTE  DTYPE              ;
                     01D1   491                 ; .BYTE  OUTLEN             ;
                     01D1   492                 ; .BYTE  JPI$C_PCBFLDTYPE   ;
                     01D1   493                 ;-----------------------------;
                     01D1   494
000001D1             01D1   495                 .BLKB   7*<MAX_PCBFLD_ITEM+1>           ; define pcbfld table
                     01D1   496
                     01D1   497 PHDFLDTBL:
                     01D1   498                 ;-----------------------------;
                     01D1   499                 ; .WORD  XXX$OFFSET         ;
```

```
                     01D1    500          ; .WORD <BITSIZ-1>@11!BITPOS  ;
                     01D1    501          ; .BYTE  DTYPE                ;
                     01D1    502          ; .BYTE  OUTLEN               ;
                     01D1    503          ; .BYTE  JPI$C_PHDFLDTYPE     ;
                     01D1    504          ;-----------------------------;
                     01D1    505
000001D1             01D1    506          .BLKB   7*<MAX_PHDFLD_ITEM+1>          ; define phdfld table
                     01D1    507
                     01D1    508          .SAVE                                  ; save current location
                     01D1    509
                     01D1    510 ;
                     01D1    511 ;************************************************************
                     01D1    512 ;
                     01D1    513 ; GENERATE THE SIX TABLES USING THE COMMONLY DEFINED MACRO
                     01D1    514 ;
                     01D1    515 ;************************************************************
                     01D1    516 ;
                     01D1    517          JPI_GENERATE_TABLE
                     01D1    518
                     01D1    519
000001D1             01D1    520          .RESTORE                               ; restore location
                     01D1    521
                     01D1    522 ;
                     01D1    523 ; Table to define items which must be handled
                     01D1    524 ; by action routines.
                     01D1    525 ;
                     01D1    526
                     01D1    527 SPECIAL:
                     01D1    528          SPECIAL_ITEM    PRI,SPC_PRI            ; handle priority ...
                     01D7    529          SPECIAL_ITEM    PRIB,SPC_PRI           ; ... evaluations
                     01DD    530          SPECIAL_ITEM    AUTHPRI,SPC_PRI        ; all of them
                     01E3    531                                                 ; compute working set
                     01E3    532          SPECIAL_ITEM    WSAUTH,SPC_WORKSET     ; ...parameters
                     01E9    533          SPECIAL_ITEM    WSQUOTA,SPC_WORKSET    ;
                     01EF    534          SPECIAL_ITEM    WSEXTENT,SPC_WORKSET   ;
                     01F5    535          SPECIAL_ITEM    WSAUTHEXT,SPC_WORKSET  ;
                     01FB    536          SPECIAL_ITEM    DFWSCNT,SPC_WORKSET    ;
                     0201    537          SPECIAL_ITEM    IMAGNAME,SPC_IMAGNAME  ; find image name
                     0207    538          SPECIAL_ITEM    MODE,SPC_MODE          ; return the process' mode
                     020D    539          SPECIAL_ITEM    PROC_INDEX,SPC_PROC_INDEX ; create a process index
                     0213    540          SPECIAL_ITEM    MASTER_PID,SPC_MASTER_PID ; return PID of master proc.
                     0219    541
0000000C             0219    542 SPECIAL_LEN = <.-SPECIAL>/6                     ; compute number of entries
```

```
0219   544          .SBTTL  SYSGETJPI - GETJPI main program
0219   545
0219   546 ;++
0219   547 ;
0219   548 ; FUNCTIONAL DESCRIPTION:
0219   549 ;
0219   550 ;       This service allows a process to receive information about itself, or
0219   551 ;       any process which it has the UIC privilege to examine.
0219   552 ;
0219   553 ; CALLING SEQUENCE:
0219   554 ;
0219   555 ;       CALLS/CALLG
0219   556 ;
0219   557 ; INPUTS:
0219   558 ;
0219   559 ;       EFN(AP) = number of the event flag to set when all of the requested
0219   560 ;                 data is valid.
0219   561 ;       PIDADR(AP) = address of a longword containing the process ID of the
0219   562 ;                    process for which the information is being requested
0219   563 ;       PRCNAM(AP) = address of a string descriptor for the process name
0219   564 ;                    of the process for which the information is requested
0219   565 ;       ITMLST(AP) = address of a list of item descriptors of the form:
0219   566 ;
0219   567 ;                       +-------------------------------+
0219   568 ;                       !  ITEM CODE   !  BUF. LENGTH   !
0219   569 ;                       +-------------------------------+
0219   570 ;                       !       BUFFER ADDRESS          !
0219   571 ;                       +-------------------------------+
0219   572 ;                       !  ADDRESS TO RETURN LENGTH     !
0219   573 ;                       +-------------------------------+
0219   574 ;
0219   575 ;       IOSB(AP) = address of a quadword I/O status block to receive final
0219   576 ;                  status
0219   577 ;       ASTADR(AP) = address of an AST routine to be called when all of the
0219   578 ;                    requested data has been supplied.
0219   579 ;       ASTPRM(AP) = 32 bit ast parameter
0219   580 ;
0219   581 ; IMPLICIT INPUTS:
0219   582 ;
0219   583 ;       none
0219   584 ;
0219   585 ; OUTPUTS:
0219   586 ;
0219   587 ;       none
0219   588 ;
0219   589 ; IMPLICIT OUTPUTS:
0219   590 ;
0219   591 ;       none
0219   592 ;
0219   593 ; ROUTINE VALUE:
0219   594 ;
0219   595 ;       SS$_NORMAL -> normal completion
0219   596 ;       SS$_ACCVIO -> ITMLST can not be read by the calling access mode,
0219   597 ;                     or the return buffer or return length word can not
0219   598 ;                     be written by the calling access mode
0219   599 ;       SS$_BADPARAM -> an invalid item identifier was supplied
0219   600 ;       SS$_IVLOGNAM -> zero or greater than maximum length process name string
```

```
                     0219   601 ;       SS$_NONEXPR -> nonexistant/deleted process or invalid process ID
                     0219   602 ;                      specified
                     0219   603 ;       SS$_NOPRIV -> calling process does not have privilege to get information
                     0219   604 ;                     about the specified process.
                     0219   605 ;
                     0219   606 ; SIDE EFFECTS:
                     0219   607 ;
                     0219   608 ;       none
                     0219   609 ;--
                     0219   610
                00000000    611         .PSECT  YEXEPAGED                   ; only entry mask in this program section
                     0000   612
               0FFC  0000   613         .ENTRY  EXE$GETJPI,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
    0214'      31    0002   614         BRW     EXE_GETJPI                  ; transfer to real procedure
                     0005   615
                00000219    616         .PSECT  YF$$SYSGETJPI               -
                     0219   617
                     0219   618 EXE_GETJPI:
5E  DB AE    DE      0219   619         MOVAL   LOCAL_SPACE(SP),SP          ; allocate local space on stack
    FC AD    D4      021D   620         CLRL    FLAGS(FP)                   ; reset the flags longword
     0769    30      0220   621         BSBW    NAMPID                      ; get PID/PCB address of desired process
    72 50    E9      0223   622         BLBC    R0,15$                      ; exit if invalid process specified
                     0226   623 ;
                     0226   624 ; Check for, and clear possible IOSB
                     0226   625 ;
51  14 AC    D0      0226   626         MOVL    IOSB(AP),R1                 ; get IOSB address
       0B    13      022A   627         BEQL    3$                          ; branch if none
                     022C   628         IFWRT   #8,(R1),29$                 ; check access to it
        00E6 31      0232   629         BRW     30$                         ; ACCVIO
          61 7C      0235   630 29$:     CLRQ    (R1)                        ; clear IOSB
                     0237   631 ;
                     0237   632 ; Check for, and clear event flag
                     0237   633 ;
53    04 AC    9A      0237   634 3$:     MOVZBL  EFN(AP),R3                  ; get event flag number
00000000'EF    16      023B   635         JSB     SCH$CLREF                   ; clear this event flag
      54 50    E9      0241   636 4$:     BLBC    R0,15$                      ; and return on errors.
                     0244   637 ;
                     0244   638 ; Validate AST, if present. Note R4 still has our PCB address, and R9
                     0244   639 ; has the PCB address of the process we want information from.
                     0244   640 ;
    18 AC    D5      0244   641         TSTL    ASTADR(AP)
       08    13      0247   642         BEQL    5$                          ; no AST to check.
    38 A4    B5      0249   643         TSTW    PCB$W_ASTCNT(R4)            ; is quota exceeded?
       03    14      024C   644         BGTR    5$                          ; nope
        00CF 31      024E   645         BRW     35$                         ; quota exceeded - return error
                     0251   646 ;
                     0251   647 ; R10 is used to count the items that are in the other process's address
                     0251   648 ; space. The accumulated size of the user buffers is kept track of
                     0251   649 ; on top of the stack. DIRCNT(FP) counts the number of PHD cells that
                     0251   650 ; were successfully fetched from another processes' header.
                     0251   651 ;
       5A    D4      0251   652 5$:     CLRL    R10                         ; no items yet
       00    DD      0253   653         PUSHL   #0                          ; no accumulated size either
    F8 AD    D4      0255   654         CLRL    DIRCNT(FP)                  ; no fetched items yet
                     0258   655 ;
                     0258   656 ; Loop through the item descriptor blocks, validating the requested item
                     0258   657 ; identifiers and moving accessible items.  A zero item identifier terminates
```

```
                     0258   658 ; the list.
                     0258   659 ;
                     0258   660
          55  10 AC   D0 0258   661          MOVL    ITMLST(AP),R5           ; get item descriptor list address
                     025C   662 6$:       IFRD    #4,(R5),10$             ; check first longword readable
                00B6   31 0262   663          BRW     30$
                     0265   664
             55   65   D0 0265   665 7$:       MOVL    (R5),R5                 ; get pointer to next chained item list
                  F2   11 0268   666          BRB     6$                      ; process it
                     026A   667
                     026A   668 10$:
             56   85   3C 026A   669          MOVZWL  (R5)+,R6                ; get buffer size
             51   85   3C 026D   670          MOVZWL  (R5)+,R1                ; get item identifier
                  63   13 0270   671          BEQL    49$                     ; done if zero, take normal exit
        FFFF 8F   51   B1 0272   672          CMPW    R1,#JPI$_CHAIN          ; is it a chained item list
                  EC   13 0277   673          BEQL    7$
                     0279   674          IFRD    #12,(R5),11$            ; check rest of this descriptor ...
                0099   31 027F   675          BRW     30$                     ; ... plus first longword of next one
             57   85   7D 0282   676 11$:      MOVQ    (R5)+,R7                ; get buffer address and return address
                  51   DD 0285   677          PUSHL   R1                      ; save R1 across accessibility check
             50   57   D0 0287   678          MOVL    R7,R0                   ; buffer address to R0
             51   56   D0 028A   679          MOVL    R6,R1                   ; and size to R1
                  53   D4 028D   680          CLRL    R3                      ; PROBE will use PSL<PRVMOD>
     00000000'EF   16   028F   681          JSB     EXE$PROBEW              ; check write accessibility of buffer
                  51 8ED0 0295   682          POPL    R1                      ; restore R1 for use by CHECKITEM
                3C 50   E9 0298   683 15$:      BLBC    R0,31$                  ; return error if inaccessible
                  55   DD 029B   684          PUSHL   R5                      ; save R5 from action routines
                02E4   30 029D   685          BSBW    CHECKITEM               ; validate identifier and get item info.
              37 50   E9 02A0   686          BLBC    R0,41$                  ; invalid item if error
    00000000'EF   59   D1 02A3   687          CMPL    R9,SCH$GL_CURPCB        ; is this for current process?
                  17   12 02AA   688          BNEQ    16$                     ; branch if not
       5E FC AD   01   E1 02AC   689          BBC     #JPI_V_NULLSWAP,FLAGS(FP),20$
                     02B1   690                                              ; branch if it's current 'full' process
                     02B1   691
                     02B1   692          CASE    R2,<-                   ; the current process is null or swap
                     02B1   693                  45$,-   ; ADR
                     02B1   694                  45$,-   ; CTL
                     02B1   695                  20$,-   ; PCB
                     02B1   696                  20$,-   ; PHD
                     02B1   697                  20$,-   ; PCBFLD
                     02B1   698                  20$,-   ; PHDFLD
                     02B1   699                  45$-    ; JIB
                     02B1   700                  >B,#1
                     02C3   701
                     02C3   702 16$:      CASE    R2,<-                   ; it isn't the current process
                     02C3   703                  45$,-   ; ADR
                     02C3   704                  18$,-   ; CTL
                     02C3   705                  20$,-   ; PCB
                     02C3   706                  17$,-   ; PHD
                     02C3   707                  20$,-   ; PCBFLD
                     02C3   708                  17$,-   ; PHDFLD
                     02C3   709                  21$-    ; JIB
                     02C3   710                  >B,#1
                     02D5   711
                  5E   11 02D5   712 49$:      BRB     50$                     ; HELPER BRANCHES
                0064   31 02D7   713 31$:      BRW     GRET
                0048   31 02DA   714 41$:      BRW     40$
```

```
                                02DD    715
1B 24 A9    12    E1            02DD    716 17$:    BBC     #PCB$V_PHDRES,PCB$L_STS(R9),18$ ; is the header resident?
                                02E2    717         ; R0 returned from MOVEPHD can be the following:
                                02E2    718         ; SS$_ACCVIO - routine MOVEIT couldn't stuff RETLEN
                                02E2    719         ; SS$_NONEXPR - got into MOVEPHD and DELPEN was set
                                02E2    720         ; SS$_NORMAL - everything fine - got the data
                                02E2    721         ; 0 - got into MOVEPHD and PHD had gone away - get with sKAST
      03EB    30                02E2    722         BSBW    MOVEPHD                 ; must be, go get the data
   10 50      E8                02E5    723         BLBS    R0,102$                 ; got it! count it.
OC    50      D1                02E8    724         CMPL    R0,#SS$_ACCVIO          ; was the retlen bad?
      51      13                02EB    725         BEQL    GRET                    ; EQL means it was
38 FC AD    01    E0            02ED    726         BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$ ; couldn't get Null/Swap? oh oh.
      50      D5                02F2    727         TSTL    R0                      ; now check for PHD no longer RES
      0C      13                02F4    728         BEQL    19$                     ; EQL means PHD no longer RES
      46      11                02F6    729         BRB     GRET                    ; whatever the error, go return it
                                02F8    730
   F8 AD      D6                02F8    731 102$:   INCL    DIRCNT(FP)              ; successful at getting it directly
      05      11                02FB    732         BRB     19$
                                02FD    733
28 FC AD    01    E0            02FD    734 18$:    BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$
                                0302    735                                         ; Null and Swapper don't have CTL reg.
      5A    D6                  0302    736 19$:    INCL    R10                     ; count up one more for sKAST later.
04 AE       56    C0            0304    737         ADDL2   R6,4(SP)                ; Add in size of user's buffer
      08    11                  0308    738         BRB     25$
1B FC AD    01    E0            030A    739 21$:    BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$
                                030F    740                                         ; Null and Swapper don't have a JIB
      0379    30                030F    741 20$:    BSBW    MOVEIT                  ; move item to user
         55 8ED0                0312    742 25$:    POPL    R5                      ; restore R5
   26 50      E9                0315    743         BLBC    R0,GRET                 ; return length not writeable
    FF4F      31                0318    744         BRW     10$                     ; back for next descriptor
                                031B    745
   50   0C    3C                031B    746 30$:    MOVZWL  #SS$_ACCVIO,R0          ; access violation
        1E    11                031E    747         BRB     GRET
                                0320    748
   50   1C    3C                0320    749 35$:    MOVZWL  #SS$_EXQUOTA,R0         ; AST quota exceeded
        19    11                0323    750         BRB     GRET
                                0325    751
   50   14    3C                0325    752 40$:    MOVZWL  #SS$_BADPARAM,R0        ; illegal item or request
        14    11                0328    753         BRB     GRET
                                032A    754
54    D8 AD    DE                032A    755 45$:    MOVAL   SCRATCH(FP),R4          ; make a 16-byte zeroed buffer
      64       7C                032E    756         CLRQ    (R4)
   08 A4       7C                0330    757         CLRQ    8(R4)
      DA       11                0333    758         BRB     20$                     ; ... through common subroutine
                                0335    759
   50   01    3C                0335    760 50$:    MOVZWL  #SS$_NORMAL,R0          ; normal return
5A    F8 AD    D1                0338    761         CMPL    DIRCNT(FP),R10          ; any items we couldn't get?
      6E       12                033C    762         BNEQ    RESCAN                  ; if so, go obtain them.
                                033E    763 ;
                                033E    764 ; Set the event flag, post the completion status, and declare a completion AST
                                033E    765 ;
      50       DD                033E    766 GRET:   PUSHL   R0                      ; save completion status
54 00000000'EF    D0            0340    767         MOVL    SCH$GL_CURPCB,R4        ; get PCB address
   51    60 A4    D0            0347    768         MOVL    PCB$L_PID(R4),R1        ; get process's PID
      52       D4                034B    769         CLRL    R2                      ; set null priority increment
   53    04 AC    D0            034D    770         MOVL    EFN(AP),R3              ; get event flag number to set
   00000000'EF    16            0351    771         JSB     SCH$POSTEF              ; set the event flag
```

```
                51   14 AC   D0  0357   772 10$:    MOVL    IOSB(AP),R1                     ; get address of IOSB
                        09   13  035B   773          BEQL    20$                             ; branch if none
                                035D   774          IFNOWRT #8,(R1),20$                     ; check if writable
                     61 6E   D0  0363   775          MOVL    (SP),(R1)                       ; store completion status
                55   18 AC   D0  0366   776 20$:    MOVL    ASTADR(AP),R5                   ; get address of AST routine
                        15   13  036A   777          BEQL    30$                             ; branch if none specified
                        54   DC  036C   778          MOVPSL  R4                              ; get PSL
          54    54   02   16   EF  036E   779          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R4,R4 ; extract previous mode
                                0373   780          $DCLAST_S (R5),ASTPRM(AP),R4            ; queue the completion AST
                        50 8ED0  0381   781 30$:    POPL    R0                              ; restore completion status
                                04  0384   782          RET                                     ; and return.
```

```
                     0385   784          .SBTTL  RESCAN - Rescan item list creating list of items in process
                     0385   785 ;++
                     0385   786 ;
                     0385   787 ; FUNCTIONAL DESCRIPTION:
                     0385   788 ;
                     0385   789 ;       Routine to obtain information that is contained in another process's
                     0385   790 ;       virtual address space.  This is accomplished by first creating a list
                     0385   791 ;       of items that are to be obtained from the other process. An AST is
                     0385   792 ;       then queued to the process to execute a routine in this service that
                     0385   793 ;       copies the desired items to a buffer in non-paged pool.  The routine
                     0385   794 ;       then queues another AST back to the requesting process to execute
                     0385   795 ;       another routine in this service to copy the items from the system
                     0385   796 ;       buffer to the requester's buffers.
                     0385   797 ;
                     0385   798 ; CALLING SEQUENCE:
                     0385   799 ;
                     0385   800 ;       Branch
                     0385   801 ;
                     0385   802 ; INPUTS:
                     0385   803 ;
                     0385   804 ;       R10 = number of items that are in other process's address space
                     0385   805 ;       R11 = PID of other process
                     0385   806 ;       (SP) = accumulated size of user buffers. A buffer of this size
                     0385   807 ;              will be allocated from nonpaged pool to hold data from
                     0385   808 ;              the target process.
                     0385   809 ;
                     0385   810 ; OUTPUTS:
                     0385   811 ;
                     0385   812 ;       none
                     0385   813 ;
                     0385   814 ; IMPLICIT OUTPUTS:
                     0385   815 ;
                     0385   816 ;       An extended AST control block is allocated and filled-in with the
                     0385   817 ;       usual AST parameters with the extension containing a list of
                     0385   818 ;       item descriptors.  A data buffer is also allocated to contain the
                     0385   819 ;       item data.
                     0385   820 ;
                     0385   821 ; ROUTINE VALUE:
                     0385   822 ;
                     0385   823 ;       none
                     0385   824 ;
                     0385   825 ; SIDE EFFECTS:
                     0385   826 ;
                     0385   827 ;       lots
                     0385   828 ;--
                     0385   829
                     0385   830          .ENABLE LOCAL_BLOCK
                     0385   831
0A 24 A4    0A   E1  0385   832 5$:      BBC     #PCB$V_SSRWAIT,PCB$L_STS(R4),7$ ; do not wait if set
                     038A   833          ENBINT                                  ; allow interrupts again
 50   0124 8F   3C  038D   834          MOVZWL  #SS$_INSFMEM,R0                 ; indicate no pool left
      AA   11       0392   835 6$:      BRB     GRET                            ; and join common exit path
                     0394   836
                     0394   837 ; There is not enough nonpaged pool. The process must be placed into resource
                     0394   838 ; wait until pool becomes available.
                     0394   839
           50   DC  0394   840 7$:      MOVPSL  R0                              ; get current PSL
```

```
                                                H 15
        6E   50   001F0000 8F   CB  0396   841          BICL3   #PSL$M_IPL,R0,(SP)      ; wait at IPL 0 to allow ASTs
                       50 03   3C  039E   842          MOVZWL  #RSN$_NPDYNMEM,R0       ; ... for some pool to be given back
                  00000000'GF  16  03A1   843          JSB     G^SCH$RWAIT             ; quota check will be repeated
                       51 56   D0  03A7   844          MOVL    R6,R1                   ;  when process executes again
                          14   11  03AA   845          BRB     8$                      ; pool available. repeat quota check
                                   03AC   846
                                   03AC   847  RESCAN:
                                   03AC   848  ;
                                   03AC   849  ; Allocate an extended AST block
                                   03AC   850  ;
                  54  00000000'EF   D0  03AC   851          MOVL    SCH$GL_CURPCB,R4        ; get current PCB address
                    51    5A   0C   C5  03B3   852          MULL3   #ACB_C_IDESC,R10,R1     ; compute size of item descriptors
                          51   34   C0  03B7   853          ADDL2   #ACB_L_ILIST,R1         ; plus header
                          51   6E   C0  03BA   854          ADDL2   (SP),R1                 ; plus buffer size
                          56   51   D0  03BD   855          MOVL    R1,R6                   ; save request size for later storage
                  00000000'GF  16  03C0   856  8$:        JSB     G^EXE$BUFFRQUOTA        ; check buffer quota
                       C9 50   E9  03C6   857          BLBC    R0,6$                   ; quit if not enough quota
                                   03C9   858  9$:        DSBINT  120$                    ; elevate IPL to SYNCH
                  00000000'GF  16  03D3   859          JSB     G^EXE$ALONONPAGED       ; now allocate the chunk of pool
                       A9 50   E9  03D9   860          BLBC    R0,5$                   ; get out if no pool available
                                   03DC   861  ;
                                   03DC   862  ; Fill-in the standard AST parameters and header information
                                   03DC   863  ;
                                   03DC   864          ENBINT                          ; allow scheduling again
                50   0080 C4   D0  03DF   865          MOVL    PCB$L_JIB(R4),R0        ; get JIB address
                  20 A0    56   C2  03E4   866          SUBL    R6,JIB$L_BYTCNT(R0)     ; adjust buffer quota
                  0C A2    5B   D0  03E8   867          MOVL    R11,ACB$L_PID(R2)       ; PID of target process
                     5B    52   D0  03EC   868          MOVL    R2,R11                  ; save address of AST block
                  08 AB    56   B0  03EF   869          MOVW    R6,ACB$W_SIZE(R11)      ; save block size for deallocation
                  0A AB    02   90  03F3   870          MOVB    #DYN$C_ACB,ACB$B_TYPE(R11) ; set block type
                           51   DC  03F7   871          MOVPSL  R1                      ; get PSL
             51   51    02   16   EF  03F9   872          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R1 ; get requester's mode
          0B AB   51    80 8F   89  03FE   873          BISB3   #<1@ACB$V_KAST>,R1,ACB$B_RMOD(R11) ; and put into block
            18 AB    082C'CF   9E  0404   874          MOVAB   W^MOVEFU,ACB$L_KAST(R11)        ; special kernel address
              10 AB    18 AC   D0  040A   875          MOVL    ASTADR(AP),ACB$L_AST(R11) ; return ast address
                       13 13   13  040F   876          BEQL    10$                     ; skip quota check if none
                     50   1C   3C  0411   877          MOVZWL  #SS$_EXQUOTA,R0         ; assume exceeded
                       38 A4   D5  0414   878          TSTL    PCB$W_ASTCNT(R4)        ; any left
                          03   12  0417   879          BNEQ    91$                     ; hop, skip to error return
                        0089   31  0419   880          BRW     35$
                       38 A4   B7  041C   881  91$:     DECW    PCB$W_ASTCNT(R4)        ; subtract from quota
                00 0B AB    06   E2  041F   882          BBSS    #ACB$V_QUOTA,ACB$B_RMOD(R11),10$ ; and record that fact in ACB
                14 AB    1C AC   D0  0424   883  10$:     MOVL    ASTPRM(AP),ACB$L_ASTPRM(R11) ; and parameter
                20 AB    04 AC   9A  0429   884          MOVZBL  EFN(AP),ACB_L_EFN(R11)  ; efn to set on return
                24 AB    14 AC   D0  042E   885          MOVL    IOSB(AP),ACB_L_IOSB(R11) ; address of possible iosb
                28 AB    60 A4   D0  0433   886          MOVL    PCB$L_PID(R4),ACB_L_OPID(R11) ; our PID
             53   00000000'EF   D0  0438   887          MOVL    CTL$GL_PHD,R3           ; get address of process header
              2C AB    00F4 C3   D0  043F   888          MOVL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R11) ; sequence number of this image
                30 AB    5A   D0  0445   889          MOVL    R10,ACB_L_COUNT(R11)    ; item count
                      1C AB   D4  0449   890          CLRL    ACB_L_DADDR(R11)        ; no data buffer yet
                                   044C   891                                          ; allocated but location not recorded
```

```
                     044C   893 ;
                     044C   894 ; Loop through the list, copying the item descriptors for items in the
                     044C   895 ; process's address space to the extended AST block.
                     044C   896 ;
                     044C   897 ; The item descriptor list will look like:
                     044C   898 ;
                     044C   899 ;       +--------------------------------+
                     044C   900 ;       ! item code     ! buffer length  !
                     044C   901 ;       +--------------------------------+
                     044C   902 ;       !    user buffer address         !
                     044C   903 ;       +--------------------------------+
                     044C   904 ;       ! address to return length       !
                     044C   905 ;       +--------------------------------+
                     044C   906 ;
                     044C   907 ; When the item list comes back from the kernel ASTs, the item code field
                     044C   908 ; is overwritten with the actual length of the source data for each item.
                     044C   909 ;
 58   34 AB   DE     044C   910          MOVAL   ACB_L_ILIST(R11),R8     ; get address of item descriptor list
 57   10 AC   D0     0450   911          MOVL    ITMLST(AP),R7           ; get address of item specifier list
                     0454   912 12$:     IFNORD  #4,(R7),30$             ; check first longword still readable
                     045A   913 ;
     56   87   3C     045A   914 15$:     MOVZWL  (R7)+,R6                ; get user buffer size
     51   87   3C     045D   915          MOVZWL  (R7)+,R1                ; get item identifier
          46   13     0460   916          BEQL    40$                     ; if zero, we're done with list.
                     0462   917          IFNORD  #12,(R7),30$            ; check still readable
FFFF 8F   51   B1     0468   918          CMPW    R1,#JPI$_CHAIN          ; is it a chained item list
          28   13     046D   919          BEQL    19$
     7E   87   7D     046F   920          MOVQ    (R7)+,-(SP)             ; get user buffer and length addresses
        010F   30     0472   921          BSBW    CHECKITEM               ; get structure type into R2
       27   50   E9   0475   922          BLBC    R0,20$                  ; make sure argument list has not changed
     50   53   D0     0478   923          MOVL    R3,R0                   ; save item length
     53   8E   7D     047B   924          MOVQ    (SP)+,R3                ; get user buffer and length addresses
 55   AA 8F   9A     047E   925          MOVZBL  #<<1@JPI$C_PCBTYPE>!-   ; create mask of types in system space
                     0482   926                    <1@JPI$C_PCBFLDTYPE>!- ;  plus address type, which is
                     0482   927                    <1@JPI$C_JIBTYPE>!-    ;  returned as zero if not for caller
                     0482   928                    <1@JPI$C_ADRTYPE>>,R5  ;
D4 55    52   E0     0482   929          BBS     R2,R5,15$               ; branch if we already got it
          5A   D7     0486   930          DECL    R10                     ; decrement item counter
          15   19     0488   931          BLSS    20$                     ; error if count goes negative
     6E   56   C2     048A   932          SUBL2   R6,(SP)                 ; subtract user buffer size from input
          10   19     048D   933          BLSS    20$                     ; error if result goes negative
     88   56   B0     048F   934          MOVW    R6,(R8)+                ; copy user buffer size
                     0492   935 ;
     88   51   B0     0492   936 18$:     MOVW    R1,(R8)+                ; copy item identifier
     88   53   7D     0495   937          MOVQ    R3,(R8)+                ; copy user buffer and length address
          C0   11     0498   938          BRB     15$                     ; and loop through till done.
                     049A   939 ;
     57   87   D0     049A   940 19$:     MOVL    (R7)+,R7                ; pointer to next item list
          B5   11     049D   941          BRB     12$                     ; go process it
                     049F   942 ;
          14   DD     049F   943 20$:     PUSHL   S^#SS$_BADPARAM         ; set bad parameters failure
          02   11     04A1   944          BRB     35$                     ;
                     04A3   945 ;
          0C   DD     04A3   946 30$:     PUSHL   S^#SS$_ACCVIO           ; set access violation failure
        008E   31     04A5   947 35$:     BRW     100$                    ;
                     04A8   948 ;
          5A   D5     04A8   949 40$:     TSTL    R10                     ; count should be zero
```

```
                     F3   12   04AA   950          BNEQ    20$                     ; error if it is not.
                     8E   D5   04AC   951          TSTL    (SP)+                   ; so should size be zero
                     EF   12   04AE   952          BNEQ    20$                     ; error if it is not.
                1C AB   58   D0   04B0   953          MOVL    R8,ACB_L_DADDR(R11)     ; fill in buffer address
                                  04B4   954 ;
                                  04B4   955 ; The AST is queued to the destination process unless it has delete or
                                  04B4   956 ; suspend pending set, or is currently suspended.
                                  04B4   957 ;
                                  04B4   958 50$:     SETIPL  120$                    ; raise IPL to synch, lock code
                     55   5B   D0   04BB   959          MOVL    R11,R5                  ; set address of AST block
                  54   0C A5   3C   04BE   960          MOVZWL  ACB$L_PID(R5),R4        ; PIX of destination process
             54   00000000'FF44   D0   04C2   961          MOVL    @SCH$GL_PCBVEC[R4],R4   ; get PCB address
                  0C A5   60 A4   D1   04CA   962          CMPL    PCB$L_PID(R4),ACB$L_PID(R5) ; see if PIDs the same
                     59   12   04CF   963          BNEQ    80$                     ; and exit if not
                  54 24 A4   01   E0   04D1   964          BBS     #PCB$V_DELPEN,PCB$L_STS(R4),80$ ; or if delete pending
                  56 24 A4   0B   E0   04D6   965          BBS     #PCB$V_SUSPEN,PCB$L_STS(R4),90$ ; or if suspend pending
                     2C A4   09   B1   04DB   966          CMPW    #SCH$C_SUSP,PCB$W_STATE(R4)
                     50   13   04DF   967          BEQL    90$                     ; process suspended, error exit
                     2C A4   0A   B1   04E1   968          CMPW    #SCH$C_SUSPO,PCB$W_STATE(R4)
                     4A   13   04E5   969          BEQL    90$                     ; or suspended out of memory
                     2C A4   02   B1   04E7   970          CMPW    #SCH$C_MWAIT,PCB$W_STATE(R4)
                     44   13   04EB   971          BEQL    90$                     ; or an indeterminately long wait state
                        52   04   D0   04ED   972          MOVL    #PRI$_TICOM,R2          ; give a big priority increment
                  AE FC AD   01   E0   04F0   973          BBS     #JPI_V_NULLSWAP,FLAGS(FP),30$ ; don't ever queue it to these!
                00000000'EF   16   04F5   974          JSB     SCH$QAST                ; queue AST to other process
                                  04FB   975 ;
                                  04FB   976 ; If process is in compute state and at a lower priority than the requesting
                                  04FB   977 ; process, boost its current priority to the requesting process's current
                                  04FB   978 ; priority.  (Required because event reporting won't normally boost a COM
                                  04FB   979 ; state process's priority).
                                  04FB   980 ;
                     2C A4   0C   B1   04FB   981          CMPW    #SCH$C_COM,PCB$W_STATE(R4) ; process in compute state?
                     06   13   04FF   982          BEQL    60$                     ;
                     2C A4   0D   B1   0501   983          CMPW    #SCH$C_COMO,PCB$W_STATE(R4) ; or compute out of memory
                     1C   12   0505   984          BNEQ    70$                     ;
             53   00000000'EF   D0   0507   985 60$:     MOVL    SCH$GL_CURPCB,R3        ; get requestor's PCB address
                  50   0B A3   90   050E   986          MOVB    PCB$B_PRI(R3),R0        ; get requestor's current priority
                  0B A4   50   91   0512   987          CMPB    R0,PCB$B_PRI(R4)        ; other process have a higher priority?
                     0B   1E   0516   988          BGEQU   70$                     ; if GEQU yes - don't boost priority
                        50   10   91   0518   989          CMPB    #16,R0                  ; will boost be into realtime priority?
                     06   1A   051B   990          BGTRU   70$                     ; if GTRU yes - don't boost priority
                00000000'EF   16   051D   991          JSB     SCH$CHSEP               ; boost other process's priority
                        50   01   3C   0523   992 70$:     MOVZWL  #SS$_NORMAL,R0          ; so far, so good.
                                  0526   993          SETIPL  #0                      ;
                                  04   0529   994          RET                     ;
                                  052A   995 ;
                                  052A   996 ; Error recovery when the process we want to send the AST to has vanished,
                                  052A   997 ; has delete pending, or is suspended; we must release both blocks
                                  052A   998 ;
                     7E   08E8 8F   3C   052A   999 80$:     MOVZWL  #SS$_NONEXPR,-(SP)      ; non-existent process
                                  05   11   052F  1000          BRB     100$
                                  0531  1001
                     7E   03A4 8F   3C   0531  1002 90$:     MOVZWL  #SS$_SUSPENDED,-(SP)    ; process is suspended
                                  0536  1003                                          ;
                        50   08 AB   3C   0536  1004 100$:    MOVZWL  ACB$W_SIZE(R11),R0      ; need to restore BYTCNT quota
                54   00000000'GF   D0   053A  1005          MOVL    G^SCH$GL_CURPCB,R4      ; to caller of $GETJPI
                     51   0080 C4   D0   0541  1006          MOVL    PCB$L_JIB(R4),R1        ; get JIB address
```

```
                20 A1    50    C0  0546  1007            ADDL2   R0,JIB$L_BYTCNT(R1)     ; and give back quota
          03 0B AB     06    E1  054A  1008            BBC     #ACB$V_QUOTA,ACB$B_RMOD(R11),105$ ; also ASTCNT if that
                38 A4  B6    054F  1009            INCW    PCB$W_ASTCNT(R4)        ; was subtracted before
                   50  5B    D0  0552  1010 105$:     MOVL    R11,R0                  ; get address of AST block
          00000000'GF  16    0555  1011            JSB     G^EXE$DEANONPAGED       ; deallocate the block
                   50 BED0   055B  1012            POPL    R0                      ; restore status
                      00     055E  1013            SETIPL  #0                      ; restore IPL to allow page faults
          08E8 8F    50    B1  0561  1014            CMPW    R0,#SS$_NONEXPR         ; is error nonexistent process?
                      04    13  0566  1015            BEQL    130$                    ; branch if yes
                   FDD3     31  0568  1016 110$:     BRW     GRET                    ;
                            056B  1017
                            056B  1018
                            056B  1019 ;
                            056B  1020 ; The preceding code must raise IPL to synchronize access to process database,
                            056B  1021 ; but since it is paged it must be locked in memory. The usage of the SETIPL
                            056B  1022 ; macro above, both raises IPL and faults the code into memory.
                            056B  1023 ;
                            056B  1024 120$:
                         08 056B  1025            .BYTE   IPL$_SYNCH              ; end of locked code region
                            056C  1026            ASSUME  <.-5$> LE 512           ; only 512 bytes can be locked
                            056C  1027            ASSUME  <.-50$> LE 512          ; only 512 bytes can be locked
                            056C  1028
                            056C  1029 ;  If process has disappeared (has already been deleted or is in a delete
                            056C  1030 ;  pending state) in the interval between selection and queuing the AST,
                            056C  1031 ;  and the initial call indicated wild card mode, then go back to the
                            056C  1032 ;  beginning of the service. Note that wild card mode is indicated by a
                            056C  1033 ;  negative number (usually -1) in the upper word of the PID argument in
                            056C  1034 ;  the caller's argument list.
                            056C  1035
             51    08 AC    D0  056C  1036 130$:     MOVL    PIDADR(AP),R1           ; Get PIDADR from argument list
                      F6    13  0570  1037            BEQL    110$                    ; If not there, can't be wild card mode
                            0572  1038            IFNORD  #2,2(R1),110$           ; Don't repeat if cannot read parameter
                   02 A1    B5  0579  1039            TSTW    2(R1)                   ; Look at wild card indicator
                      EA    18  057C  1040            BGEQ    110$                    ; Must be negative for wild card mode
                5E    5D    D0  057E  1041            MOVL    FP,SP                   ; Restore SP to its value on entry
                   FA7E'    31  0581  1042            BRW     EXE$GETJPI + 2          ;  and go back to the beginning.
                            0584  1043
                            0584  1044            .DISABLE        LOCAL_BLOCK
                            0584  1045
```

```
0584  1047          .SBTTL  CHECKITEM - Validate item identifier
0584  1048
0584  1049 ;++
0584  1050 ;
0584  1051 ; FUNCTIONAL DESCRIPTION:
0584  1052 ;
0584  1053 ;       Routine to validate item identifier and return information
0584  1054 ;       about the item.
0584  1055 ;
0584  1056 ; CALLING SEQUENCE:
0584  1057 ;
0584  1058 ;       JSB/BSB
0584  1059 ;
0584  1060 ; INPUTS:
0584  1061 ;
0584  1062 ;       R1 = item identifier
0584  1063 ;       R9 = Target PCB address
0584  1064 ;
0584  1065 ; IMPLICIT INPUTS:
0584  1066 ;
0584  1067 ;       none
0584  1068 ;
0584  1069 ; OUTPUTS:
0584  1070 ;
0584  1071 ;       R1 = item identifier
0584  1072 ;       R2 = structure number
0584  1073 ;       R3 = item length
0584  1074 ;       R4 = item address (actual address for PCB data, assumes current process
0584  1075 ;               for other data) < if we're getting PHD data directly, it will
0584  1076 ;                                be the PHD offset, not the address >
0584  1077 ;       R5 = item type code
0584  1078 ;
0584  1079 ; IMPLICIT OUTPUTS:
0584  1080 ;
0584  1081 ;       none
0584  1082 ;
0584  1083 ; ROUTINE VALUE:
0584  1084 ;
0584  1085 ;       R0 low bit clear -> successful return
0584  1086 ;       R0 low bit set -> invalid item identifier
0584  1087 ;
0584  1088 ; SIDE EFFECTS:
0584  1089 ;
0584  1090 ;       none
0584  1091 ;--
```

```
                         0584  1093 CHECKITEM:
                   50    D4  0584  1094          CLRL    R0                          ; assume error
                53 51    9A  0586  1095          MOVZBL  R1,R3                       ; get item number
          52 51 08       EF  0589  1096          EXTZV   #8,#8,R1,R2                 ; get structure number
                1F       13  058E  1097          BEQL    79$                         ; error if structure number zero
                06 52    91  0590  1098          CMPB    R2,#JPI$C_MAXSTRUC          ; structure number valid?
                1A       1A  0593  1099          BGTRU   79$                         ; error if not
             FA64 CF42   53  91  0595  1100          CMPB    R3,MAXCOUNT-1[R2]           ; check max item values (1 origin)
                12       1A  059B  1101          BGTRU   79$                         ; error if illegal item number
                54       D4  059D  1102          CLRL    R4                          ; assume zero base
                         059F  1103          CASE    R2,<-                       ; case on structure base
                         059F  1104                      10$,-           ; ADR
                         059F  1105                      50$,-           ; CTL
                         059F  1106                      20$,-           ; PCB
                         059F  1107                      30$,-           ; PHD
                         059F  1108                      100$,-          ; PCBFLD
                         059F  1109                      110$-           ; PHDFLD
                         059F  1110                      >B,#1
                         05AF  1111
                         05AF  1112 79$:
                0088     31  05AF  1113          BRW     80$                         ; CASE out of bounds - return
                         05B2  1114
                         05B2  1115 10$:                                      ; item is an address
             54 FA4F CF43    DE  05B2  1116          MOVAL   ADRTBL[R3],R4               ; address is table address
                   54 53     C0  05B8  1117          ADDL    R3,R4                       ; base+indexvalue*5
                52    04 A4  9A  05BB  1118          MOVZBL  4(R4),R2                    ; get structure type code
                   53 04     D0  05BF  1119          MOVL    #4,R3                       ; size of data is four bytes
                   55 00     9A  05C2  1120          MOVZBL  #VALUE,R5                   ; item is a value
                      00A3   31  05C5  1121          BRW     70$                         ; all done
                         05C8  1122 20$:                                      ; item is from PCB
                   54 59     D0  05C8  1123          MOVL    R9,R4                       ; get back PCB address
                55 FAB8 CF    DE  05CB  1124          MOVAL   PCBTBL,R5                   ; get address of PCB item table
                      15     11  05D0  1125          BRB     40$                         ; continue
                         05D2  1126 30$:                                      ; item is from process header
00000000'EF      59     D1  05D2  1127          CMPL    R9,SCH$GL_CURPCB            ; is the target process our own?
                   07     12  05D9  1128          BNEQ    35$                         ; NEQ means it's not, don't touch CTL
54 00000000'9F    D0  05DB  1129          MOVL    @#CTL$GL_PHD,R4             ; get process header address
55 FB5F CF        DE  05E2  1130 35$:    MOVAL   PHDTBL,R5                   ; get address of PHD item table
                         05E7  1131 40$:
                53 05     C4  05E7  1132          MULL    #5,R3                       ; each element is 5 bytes long
                53 55     C0  05EA  1133          ADDL    R5,R3                       ; compute address in item table
                55 83     3C  05ED  1134          MOVZWL  (R3)+,R5                    ; get offset into data structure
                         05F0  1135
                      3E     11  05F0  1136          BRB     60$                         ;
                         05F2  1137 100$:                                     ; item is from PCBFLD
                54 59     D0  05F2  1138          MOVL    R9,R4                       ; get back PCB address
                53 07     C4  05F5  1139          MULL    #7,R3                       ; each element is 7 bytes long
          53 FBD4 CF43    9E  05F8  1140          MOVAB   PCBFLDTBL[R3],R3            ; get address of PCBFLD item
                      19     11  05FE  1141          BRB     120$                        ; continue
                         0600  1142 110$:                                     ; item is from PHDFLD
00000000'EF      59     D1  0600  1143          CMPL    R9,SCH$GL_CURPCB            ; is the target process our own?
                   07     12  0607  1144          BNEQ    115$                        ; NEQ means not, don't touch CTL
54 00000000'9F    D0  0609  1145          MOVL    @#CTL$GL_PHD,R4             ; get process header address
                53 07     C4  0610  1146 115$:   MULL    #7,R3                       ; each element is 7 bytes long
          53 FBB9 CF43    9E  0613  1147          MOVAB   PHDFLDTBL[R3],R3            ; get address of PHDFLD item
                         0619  1148 120$:
                55 83     3C  0619  1149          MOVZWL  (R3)+,R5                    ; get offset into data structure
```

```
                E8 AD    83    3C  061C  1150          MOVZWL  (R3)+,BITDEF(FP)         ; save the BITSIZ and BITPOS
                                    0620  1151
                         0E    11  0620  1152          BRB     60$                     ;
                                    0622  1153  50$:                                  ; item is in control region
                   53    07    C4  0622  1154          MULL    #7,R3                   ; compute index into item table
                         54    D4  0625  1155          CLRL    R4                      ; assume zero base value
             53    F9E4 CF43    9E  0627  1156          MOVAB   CTLTBL[R3],R3           ; get address of item information
                   55    83    D0  062D  1157          MOVL    (R3)+,R5                ; get item address
                52    02    A3    9A  0630  1158  60$:   MOVZBL  2(R3),R2               ; fetch actual structure type
                   52    07    91  0634  1159          CMPB    #JPI$C_JIBTYPE,R2       ; is it the JIB?
                         05    12  0637  1160          BNEQ    65$                     ; br if not
                54    0080  C9    D0  0639  1161          MOVL    PCB$L_JIB(R9),R4        ; else get address of JIB
                   54    55    C0  063E  1162  65$:     ADDL    R5,R4                   ; form complete address
                   55    83    9A  0641  1163          MOVZBL  (R3)+,R5                ; get item type code
                   55    04    D1  0644  1164          CMPL    #DSTRING,R5             ; is it a string descriptor?
                         06    12  0647  1165          BNEQ    67$                     ; NEQ means nope
                FC AD    04    C8  0649  1166          BISL2   #<1@JPI_V_STRDSC>,FLAGS(FP) ; it's special, flag it
                         04    11  064D  1167          BRB     69$
                FC AD    04    CA  064F  1168  67$:     BICL2   #<1@JPI_V_STRDSC>,FLAGS(FP) ; not special, clear flag
                   53    63    9A  0653  1169  69$:     MOVZBL  (R3),R3                 ; get item length
                   52    05    91  0656  1170          CMPB    #JPI$C_PCBFLDTYPE,R2    ; is it a bit field?
                         0E    13  0659  1171          BEQL    90$                     ; EQL means it is
                   52    06    91  065B  1172          CMPB    #JPI$C_PHDFLDTYPE,R2    ; is it s bit field?
                         0B    12  065E  1173          BNEQ    70$                     ; NEQ means it's not a FLD at all
00000000'EF              59    D1  0660  1174          CMPL    R9,SCH$GL_CURPCB        ; is the target process our own?
                         02    12  0667  1175          BNEQ    70$                     ; NEQ means it's not
                         03    10  0669  1176  90$:     BSBB    EXTFLD
                         50    D6  066B  1177  70$:     INCL    R0                      ; set successful return
                               05  066D  1178  80$:     RSB                             ; return to caller
```

```
                                066E  1180          .SBTTL  EXTFLD - Extract a bitfield from a datum
                                066E  1181
                                066E  1182 ;++
                                066E  1183 ;
                                066E  1184 ; FUNCTIONAL DESCRIPTION:
                                066E  1185 ;
                                066E  1186 ;       Routine to fetch bitfield data from within a data cell.
                                066E  1187 ;
                                066E  1188 ; CALLING SEQUENCE:
                                066E  1189 ;
                                066E  1190 ;       JSB/BSB
                                066E  1191 ;
                                066E  1192 ; INPUTS:
                                066E  1193 ;
                                066E  1194 ;       BITDEF(FP) = BITPOS/BITSIZ fields from item table
                                066E  1195 ;       R4 = address of cell containing data
                                066E  1196 ;
                                066E  1197 ; IMPLICIT INPUTS:
                                066E  1198 ;
                                066E  1199 ;       none
                                066E  1200 ;
                                066E  1201 ; OUTPUTS:
                                066E  1202 ;
                                066E  1203 ;       R4 = new address on stack where bit is saved
                                066E  1204 ;
                                066E  1205 ; IMPLICIT OUTPUTS:
                                066E  1206 ;
                                066E  1207 ;       none
                                066E  1208 ;
                                066E  1209 ; ROUTINE VALUE:
                                066E  1210 ;
                                066E  1211 ;       none
                                066E  1212 ;
                                066E  1213 ; SIDE EFFECTS:
                                066E  1214 ;
                                066E  1215 ;       none
                                066E  1216 ;--
                                066E  1217
                                066E  1218 EXTFLD:
                          0C    BB  066E  1219          PUSHR   #^M<R2,R3>                      ; get some room
52    E8 AD    05         0B    EF  0670  1220          EXTZV   #11,#5,BITDEF(FP),R2            ; get BITSIZ-1
                          52    D6  0676  1221          INCL    R2                              ; make it BITSIZ
53    E8 AD    0B         00    EF  0678  1222          EXTZV   #0,#11,BITDEF(FP),R3            ; get BITPOS
E8 AD    64    52         53    EF  067E  1223          EXTZV   R3,R2,(R4),BITDEF(FP)           ; get the bitfield
            54    E8 AD        DE  0684  1224          MOVAL   BITDEF(FP),R4                   ; point to it
                          0C    BA  0688  1225          POPR    #^M<R2,R3>                      ; restore the registers
                          05    05  068A  1226          RSB
```

```
                        .SBTTL  MOVEIT - Move data to user's buffer

;++
;
; FUNCTIONAL DESCRIPTION:
;
;       Move the requested data to user buffer. Zero fill to end of buffer.
;       Return actual data length to user.  Assumes user's buffer has
;       been probed.
;
; CALLING SEQUENCE:
;
;       JSB/BSB
;
; INPUTS:
;
;       R1 = item identifier
;       R2 = data structure number
;       R3 = item length
;       R4 = item address
;       R5 = item type code
;       R6 = user buffer length
;       R7 = user buffer address
;       R8 = address to return length
;       R11 = PID of process to get data from
;
; IMPLICIT INPUTS:
;
;       none
;
; OUTPUTS:
;
;       none
;
; IMPLICIT OUTPUTS:
;
;       none
;
; ROUTINE VALUE:
;
;       R0 low bit set -> success
;       R0 low bit clear -> access violation on write of length
;
; SIDE EFFECTS:
;
;       Registers R1-R4 destroyed
;--
```

```
                                                D 16
                                0688  1276 MOVEIT:
                                0688  1277
                                0688  1278 ;
                                0688  1279 ; Call routine to check for special conditions
                                0688  1280 ;
                                0688  1281
                        00A0  30 0688  1282         BSBW    CHECK_SPC
                                068E  1283
                                068E  1284 ;
                                068E  1285 ; Check for counted string, and find actual length if so.
                                068E  1286 ;
                                068E  1287
                     55  02  D1 068E  1288         CMPL    #CSTRING,R5             ; is this special string?
                         03  12 0691  1289         BNEQ    10$                     ; branch if not
                     53  84  9A 0693  1290         MOVZBL  (R4)+,R3                ; get length and skip length byte
                                0696  1291 ;
                                0696  1292 ; Check that process still exists.  This assures that data address is good.
                                0696  1293 ;
                     50  5B  3C 0696  1294 10$:     MOVZWL  R11,R0                  ; get process ID index
           50  00000000'FF40  D0 0699  1295         MOVL    @SCH$GL_PCBVEC[R0],R0   ; get PCB address
                  5B  60 A0  D1 06A1  1296         CMPL    PCB$L_PID(R0),R11       ; same PID?
                         02  13 06A5  1297         BEQL    15$                     ; branch if yes
                         53  D4 06A7  1298         CLRL    R3                      ; else, zero data size
                                06A9  1299 ;
                                06A9  1300 ; Move the data
                                06A9  1301 ;
                         28  BB 06A9  1302 15$:     PUSHR   #^M<R3,R5>              ; save needed registers from movc
           67  56  00  64  53  2C 06AB  1303         MOVC5   R3,(R4),#0,R6,(R7)      ; move data to user's buffer, zero fill
                         28  BA 06B1  1304         POPR    #^M<R3,R5>              ; restore registers
                         58  D5 06B3  1305         TSTL    R8                      ; did caller want return length?
                         11  13 06B5  1306         BEQL    30$                     ; branch if not
                                06B7  1307         IFNOWRT #2,(R8),40$             ; exit if word not writable
                     56  53  D1 06BD  1308         CMPL    R3,R6                   ; see how much was moved
                         03  15 06C0  1309         BLEQ    20$                     ; use valid data length if it fit
                     53  56  D0 06C2  1310         MOVL    R6,R3                   ; else give him "too short" buffer size
                     68  53  B0 06C5  1311 20$:     MOVW    R3,(R8)                 ; return length to user
                     50  01  3C 06C8  1312 30$:     MOVZWL  #SS$_NORMAL,R0          ; set success code
                                05 06CB  1313         RSB
                                06CC  1314
                     50  0C  3C 06CC  1315 40$:     MOVZWL  #SS$_ACCVIO,R0          ; couldn't stuff RETLEN cell
                                05 06CF  1316         RSB                             ; return
```

```
                        06D0  1318          .SBTTL  MOVEPHD - Fetch data from other process' PHD
                        06D0  1319
                        06D0  1320 ;++
                        06D0  1321 ;
                        06D0  1322 ; FUNCTIONAL DESCRIPTION:
                        06D0  1323 ;
                        06D0  1324 ;       Disable interrupts and fetch the data from the other process' header.
                        06D0  1325 ;       Always get WSLIST in case it'll be needed. Make R4 point to the
                        06D0  1326 ;       saved data, and call EXTFLD if it's a PHDFLDTYPE item.
                        06D0  1327 ;
                        06D0  1328 ; CALLING SEQUENCE:
                        06D0  1329 ;
                        06D0  1330 ;       JSB/BSB
                        06D0  1331 ;
                        06D0  1332 ; INPUTS:
                        06D0  1333 ;
                        06D0  1334 ;       R1 = item identifier
                        06D0  1335 ;       R2 = data structure number
                        06D0  1336 ;       R3 = item length
                        06D0  1337 ;       R4 = offset into other process' PHD
                        06D0  1338 ;       R5 = item type code
                        06D0  1339 ;       R6 = user buffer length
                        06D0  1340 ;       R7 = user buffer address
                        06D0  1341 ;       R8 = address to return length
                        06D0  1342 ;       R11 = PID of process to get data from
                        06D0  1343 ;
                        06D0  1344 ; IMPLICIT INPUTS:
                        06D0  1345 ;
                        06D0  1346 ;       none
                        06D0  1347 ;
                        06D0  1348 ; OUTPUTS:
                        06D0  1349 ;
                        06D0  1350 ;       none
                        06D0  1351 ;
                        06D0  1352 ; IMPLICIT OUTPUTS:
                        06D0  1353 ;
                        06D0  1354 ;       none
                        06D0  1355 ;
                        06D0  1356 ; ROUTINE VALUE:
                        06D0  1357 ;
                        06D0  1358 ;       SS$_ACCVIO - routine MOVEIT couldn't stuff RETLEN
                        06D0  1359 ;       SS$_NONEXPR - got into MOVEPHD and DELPEN was set
                        06D0  1360 ;       SS$_NORMAL - everything fine - got the data
                        06D0  1361 ;       0 - got into MOVEPHD and PHD had gone away - get with sKAST
                        06D0  1362 ;
                        06D0  1363 ; SIDE EFFECTS:
                        06D0  1364 ;
                        06D0  1365 ;       Registers R1-R4 destroyed
                        06D0  1366 ;--
                        06D0  1367
                        06D0  1368 MOVEPHD:
                  7E    55    7D  06D0  1369          MOVQ    R5,-(SP)                ; save R5 and R6
                        06D3  1370 LOCK_BEGIN = .
                        06D3  1371          DSBINT  LOCK_IPL                ; raise IPL to Synch and lock code
                  50    5B    3C  06DD  1372          MOVZWL  R11,R0                  ; get process ID index
              50  00000000'FF40   D0  06E0  1373          MOVL    @SCH$GL_PCBVEC[R0],R0   ; get PCB address
                  5B    60 A0   D1  06E8  1374          CMPL    PCB$L_PID(R0),R11       ; same PID?
```

```
                37   12   06EC  1375          BNEQ    90$                             ; NEQ means not the same
32 24 A0        12   E1   06EE  1376          BBC     #PCB$V_PHDRES,PCB$L_STS(R0),90$ ; if the PHD isn't there, exit
26 24 A0        01   E0   06F3  1377          BBS     #PCB$V_DELPEN,PCB$L_STS(R0),85$ ; if process will go away, exit
   55    6C A0   D0   06F8  1378          MOVL    PCB$L_PHD(R0),R5                ; get the header address
F4 AD    08 A5   3C   06FC  1379          MOVZWL  PHD$W_WSLIST(R5),WSLIST(FP) ; save the WSLIST just in case
56    55    54   C1   0701  1380          ADDL3   R4,R5,R6                        ; PHD offset + PHD address => R6
   EC AD    66   7D   0705  1381          MOVQ    (R6),PHDTEMP(FP)                ; save the data from the PHD
                       0709  1382          ENBINT                                  ; allow interrupts again
   54    EC AD   DE   070C  1383          MOVAL   PHDTEMP(FP),R4                  ; point to the saved data
      55    8E   7D   0710  1384          MOVQ    (SP)+,R5                        ; restore R5 and R6
      52    06   91   0713  1385          CMPB    #JPI$C_PHDFLDTYPE,R2            ; is it a bit field?
            03   12   0716  1386          BNEQ    30$                             ; NEQ means it is not
         FF53   30   0718  1387          BSBW    EXTFLD                          ; extract out the bitfield
         FF6D   31   071B  1388 30$:     BRW     MOVEIT                          ; now "fetch" the data the normal way
                     071E  1389
50    08E8 8F   3C   071E  1390 85$:     MOVZWL  #SS$_NONEXPR,R0                 ; process going away
            02   11   0723  1391          BRB     95$
                     0725  1392
            50   D4   0725  1393 90$:     CLRL    R0                              ; PHD not resident anymore
                     0727  1394 95$:     ENBINT
      55    8E   7D   072A  1395          MOVQ    (SP)+,R5                        ; clean off the stack, restore R5,R6
                  05   072D  1396          RSB
                       072E  1397
```

```
                     072E  1399          .SBTTL  SPECIAL - Handle special conditions
                     072E  1400
                     072E  1401 ;++
                     072E  1402 ;
                     072E  1403 ; FUNCTIONAL DESCRIPTION:
                     072E  1404 ;
                     072E  1405 ;       These routines handle data items which must be transformed
                     072E  1406 ;       before they are returned to the user.  Generally, some
                     072E  1407 ;       transformation is applied to the data item and the newly
                     072E  1408 ;       computed item is stored in LOCAL_SPACE on the stack.
                     072E  1409 ;       The handling routine then changes R4 to point to LOCAL_SPACE
                     072E  1410 ;       so that MOVEIT will move the item from local storage.
                     072E  1411 ;
                     072E  1412 ; CALLING SEQUENCE:
                     072E  1413 ;
                     072E  1414 ;       JSB/BSB
                     072E  1415 ;
                     072E  1416 ; INPUTS:
                     072E  1417 ;
                     072E  1418 ;       R1 = item identifier
                     072E  1419 ;       R3 = item length
                     072E  1420 ;       R4 = item address
                     072E  1421 ;       R6 = user's buffer length
                     072E  1422 ;       R9 = PCB address of target process
                     072E  1423 ;
                     072E  1424 ; IMPLICIT INPUTS:
                     072E  1425 ;
                     072E  1426 ;       none
                     072E  1427 ;
                     072E  1428 ; OUTPUTS:
                     072E  1429 ;
                     072E  1430 ;       none
                     072E  1431 ;
                     072E  1432 ; IMPLICIT OUTPUTS:
                     072E  1433 ;
                     072E  1434 ;       none
                     072E  1435 ;
                     072E  1436 ; ROUTINE VALUE:
                     072E  1437 ;
                     072E  1438 ;       none
                     072E  1439 ;
                     072E  1440 ; SIDE EFFECTS:
                     072E  1441 ;
                     072E  1442 ;       none
                     072E  1443 ;--
                     072E  1444
                     072E  1445 CHECK_SPC:
                     072E  1446
                     072E  1447 ;
                     072E  1448 ; Registers R5 and R6 are saved at this level and may be used by
                     072E  1449 ; the action routines without being saved.  Action routines are JSB'ed
                     072E  1450 ; to with R5 containing the address of LOCAL_SPACE on the stack.
                     072E  1451 ;
                     072E  1452
     7E    55   7D  072E  1453          MOVQ    R5,-(SP)                ; save registers
     55    0C   D0  0731  1454          MOVL    #SPECIAL_LEN,R5         ; get number of table entries
56   FA99  CF   DE  0734  1455          MOVAL   SPECIAL,R6              ; get address of table
```

```
                                                H 16
                     0739  1456 10$:
                 86  51  B1  0739  1457          CMPW    R1,(R6)+                ; does entry match item?
                     08  13  073C  1458          BEQL    20$                     ; yes, go handle it
                 56  04  C0  073E  1459          ADDL    #4,R6                   ; skip handler address
                   F5  55  F5  0741  1460          SOBGTR  R5,10$                  ; scan rest of table
                     06  11  0744  1461          BRB     30$                     ; nothing to do, exit
                     0746  1462 20$:
              55     DB  AD  DE  0746  1463          MOVAL   LOCAL_SPACE(FP),R5      ; load local address for action routine
                     96  16  074A  1464          JSB     @(R6)+                  ; call action routine
                     074C  1465 30$:
                 55  8E  7D  074C  1466          MOVQ    (SP)+,R5                ; restore registers
                     05  074F  1467          RSB
                     0750  1468 ;
                     0750  1469 ; Data handling routines
                     0750  1470 ;
                     0750  1471
                     0750  1472 ;
                     0750  1473 ; Internal priority must be subtracted from 31 before being returned.
                     0750  1474 ;
                     0750  1475
                     0750  1476 SPC_PRI:
                65   1F  64  83  0750  1477          SUBB3   (R4),#31,(R5)           ; compute external priority
                     54  55  D0  0754  1478          MOVL    R5,R4                   ; change address for move routine
                     05  0757  1479          RSB
                     0758  1480
                     0758  1481 ;
                     0758  1482 ; Check the PCB$L_STS bits to get the mode
                     0758  1483 ;
                     0758  1484
                     0758  1485 SPC_MODE:
                 56   24  A9  D0  0758  1486          MOVL    PCB$L_STS(R9),R6        ; the bits are in the STS
                   65    01  D0  075C  1487          MOVL    #JPI$K_NETWORK,(R5)     ; assume network
                 11  56    15  E0  075F  1488          BBS     #PCB$V_NETWRK,R6,10$    ; if set, all done
                   65    02  D0  0763  1489          MOVL    #JPI$K_BATCH,(R5)       ; now try batch mode
                 0A  56    0E  E0  0766  1490          BBS     #PCB$V_BATCH,R6,10$
                   65    03  D0  076A  1491          MOVL    #JPI$K_INTERACTIVE,(R5) ; now try interactive mode
                 03  56    19  E0  076D  1492          BBS     #PCB$V_INTER,R6,10$
                   65    00  D0  0771  1493          MOVL    #JPI$K_OTHER,(R5)       ; it must be 'other' mode
                     54  55  D0  0774  1494 10$:      MOVL    R5,R4                   ; point at the 'data'
                     05  0777  1495          RSB
                     0778  1496
                     0778  1497 ;
                     0778  1498 ; Working set pointers are indices into working set list
                     0778  1499 ; and must be subtracted from first list element.
                     0778  1500 ;
                     0778  1501
                     0778  1502 SPC_WORKSET:
00000000'EF          59  D1  0778  1503          CMPL    R9,SCH$GL_CURPCB        ; is the target process our own?
                     07  13  077F  1504          BEQL    15$                     ; EQL means it is
 65  64  F4 AD      A3  0781  1505          SUBW3   WSLIST(FP),(R4),(R5)    ; use the saved WSLIST
                     0C  11  0786  1506          BRB     17$                     ; don't touch CTL here
56   00000000'9F    D0  0788  1507 15$:      MOVL    @#CTL$GL_PHD,R6         ; get process header address
 65  64    08 A6    A3  078F  1508          SUBW3   PHD$W_WSLIST(R6),(R4),(R5) ; compute argument size
           54  55    D0  0794  1509 17$:      MOVL    R5,R4                   ; change item address
               64    B6  0797  1510          INCW    (R4)                    ; must add one to index
                     05  0799  1511          RSB
                     079A  1512
```

```
                                                                 ; 16
                        079A  1513 ;
                        079A  1514 ; Convert the pcb vector index into a process index.  This is so that applications
                        079A  1515 ; which used to use the low word of the PID as an index can adapt.  This item,
                        079A  1516 ; JPI$_PROC_INDEX, is very similar to the current pix:
                        079A  1517 ;
                        079A  1518 ;       PROC_INDEX is a number between 1 and the sysgen parameter MAXPROCESSCNT.
                        079A  1519 ;       This means that it is a small number, and that it is practical to
                        079A  1520 ;       statically allocate bitvectors or other vectors for the maximum
                        079A  1521 ;       number of processes expected.
                        079A  1522 ;
                        079A  1523 ;       At any instant, no more than one process will have a given PROC_INDEX.
                        079A  1524 ;       In particular, no other process will have the same PROC_INDEX as you.
                        079A  1525 ;       This guarantees no collisions.  If the application wanted to know
                        079A  1526 ;       about reuse of the PROC_INDEX, it could store the EPID in the
                        079A  1527 ;       vector and do its own check.
                        079A  1528 ;
                        079A  1529 ; No program should assume that PROC_INDEX is anything more than this.  Note that
                        079A  1530 ; the sole purpose of the following arithmetic is to make sure that PROC_INDEX is
                        079A  1531 ; NOT the pcb vector index!
                        079A  1532 ;
                        079A  1533 SPC_PROC_INDEX:
          56   64   3C  079A  1534         MOVZWL  (R4),R6                 ; Get the PIX into a register
65  00000000'EF   3C  079D  1535         MOVZWL  SGN$GW_MAXPRCCT,(R5)    ; Move the MAXPROCESSCNT into the scratch
          65   56   C2  07A4  1536         SUBL2   R6,(R5)                 ; Convert 0 to N-1 to range N to 1
          54   55   D0  07A7  1537         MOVL    R5,R4                   ; Point at the new value for move routine
                    05  07AA  1538         RSB
                        07AB  1539
                        07AB  1540
                        07AB  1541 ;
                        07AB  1542 ; Convert the MPID from the JIB to extended format.
                        07AB  1543 ;
                        07AB  1544 ; Inputs:
                        07AB  1545 ;
                        07AB  1546 ;       R4 = Addr of MPID in internal format
                        07AB  1547 ;       R5 = Addr of scratch buffer
                        07AB  1548 ;
                        07AB  1549 ; Outputs:
                        07AB  1550 ;
                        07AB  1551 ;       R4 = Addr. MPID in extended format
                        07AB  1552 ;
                        07AB  1553
                        07AB  1554 SPC_MASTER_PID:
          50   64   D0  07AB  1555         MOVL    (R4),R0                 ; get MPID
00000000'GF   16   07AE  1556         JSB     G^EXE$IPID_TO_EPID      ; convert it to extended format
          65   50   D0  07B4  1557         MOVL    R0,(R5)                 ; store converted PID in scratch buffer
          54   55   D0  07B7  1558         MOVL    R5,R4                   ; point to the converted PID
                    05  07BA  1559         RSB
```

```
                     07B8  1561 ;
                     07B8  1562 ; The current image file name is in the Image File Descriptor Block.  It
                     07B8  1563 ; is also in user writable memory, so all addresses must be probed.
                     07B8  1564 ;
                     07B8  1565 ; Inputs:
                     07B8  1566 ;       R4 = CTL$GL_IMGHDRBF, address of image header buffer
                     07B8  1567 ;
                     07B8  1568 ; Outputs:
                     07B8  1569 ;       R3 = size of image file name
                     07B8  1570 ;       R4 = address of image file name
                     07B8  1571 ;
                     07B8  1572
                     07B8  1573 EXE$CHKIMAGNAME::
    54    64    D0   07B8  1574         MOVL    (R4),R4                 ; get address of image header buffer
          19    13   07BB  1575         BEQL    11$                     ; if EQL, no image active
55    04 A4    D0   07C0  1576         MOVL    4(R4),R5                ; get address of image file descriptor
                     07C4  1577         IFNORD  #8,IFD$Q_CURPROG(R5),11$,#PSL$C_USER ; check access to desc
53    14 A5    7D   07CB  1578         MOVQ    IFD$Q_CURPROG(R5),R3    ; get image name descriptor
   53    53    3C   07CF  1579         MOVZWL  R3,R3                   ; assure size of string is in range
                     07D2  1580         IFNORD  R3,(R4),11$,#PSL$C_USER ; check access to string
                05   07D8  1581         RSB                             ;
                     07D9  1582 11$:
          53    7C   07D9  1583         CLRQ    R3                      ; zero string descriptor
                05   07DB  1584         RSB                             ;
                     07DC  1585
                     07DC  1586 ;
                     07DC  1587 ; The current image file name is in the Image File Descriptor Block.
                     07DC  1588 ; Probe it for maximum protection.
                     07DC  1589 ;
                     07DC  1590 ; If a compatibility mode exception handler has been declared for the
                     07DC  1591 ; proc      ssume that an AME is running. Further assume that the second
                     07DC  1592 ; compat...ity mode context page has been patterned enough after the
                     07DC  1593 ; image file descriptor block such that an alternate image name can be
                     07DC  1594 ; found there. In this case, return that image name. If the name is null,
                     07DC  1595 ; fall back to the name in the Image File Descriptor Block. Note that the
                     07DC  1596 ; second compatibility mode context page is user writeable, so it must
                     07DC  1597 ; be probed.
                     07DC  1598 ;
                     07DC  1599 ; Inputs:
                     07DC  1600 ;       R4 = CTL$GL_IMGHDRBF, address of image header buffer
                     07DC  1601 ;       8(SP) = user's buffer length
                     07DC  1602 ;
                     07DC  1603 ; Outputs:
                     07DC  1604 ;       R3 = size of image file name
                     07DC  1605 ;       R4 = address of image file name
                     07DC  1606 ;
                     07DC  1607
                     07DC  1608 SPC_IMAGNAME:
    56    08 AE    D0   07DC  1609         MOVL    8(SP),R6                ; get the user's buffer length
          54    64   D0   07E0  1610         MOVL    (R4),R4                 ; get address of image header buffer
                43   13   07E3  1611         BEQL    10$                     ; if EQL, no image active
          00000000'GF   D5   07E5  1612         TSTL    G^CTL$GL_CMHANDLR       ; is there an AME running?
                1A   13   07EB  1613         BEQL    5$                      ; if EQL no use image in IFD
55        00000200'GF   DE   07ED  1614         MOVAL   G^CTL$AL_CMCNTX+^X200,R5 ; point to second c-mode context page
                     07F4  1615         IFNORD  #8,IFD$Q_CURPROG(R5),10$,#PSL$C_USER ; check access to desc
    53    14 A5    D0   07FB  1616         MOVL    IFD$Q_CURPROG(R5),R3    ; get length of image name string
                06   13   07FF  1617         BEQL    5$                      ; if EQL, string is null, get from IFD
```

```
                                                    K 16
54   18 A5   D0  0801  1618          MOVL    IFD$Q_CURPROG+4(R5),R4  ; get address of string
        0F   11  0805  1619          BRB     6$                      ; join common code
                 0807  1620
55   04 A4   D0  0807  1621 5$:      MOVL    4(R4),R5                ; get address of image file descriptor
                 080B  1622          IFNORD  #8,IFD$Q_CURPROG(R5),10$,#PSL$C_USER ; check access to desc
53   14 A5   7D  0812  1623          MOVQ    IFD$Q_CURPROG(R5),R3    ; get image name descriptor
   53   53   3C  0816  1624 6$:      MOVZWL  R3,R3                   ; assure size of string is in range
   56   53   D1  0819  1625          CMPL    R3,R6                   ; is it bigger than the user's buffer?
        03   15  081C  1626          BLEQ    7$
   53   56   D0  081E  1627          MOVL    R6,R3                   ; yes, make user's size the real size
                 0821  1628 7$:      IFNORD  R3,(R4),10$,#PSL$C_USER ; check access to string
             05  0827  1629          RSB                             ;
                 0828  1630 10$:
        53   7C  0828  1631          CLRQ    R3                      ; zero string descriptor
             05  082A  1632          RSB                             ;
                 082B  1633
                 082B  1634 LOCK_IPL:
             08  082B  1635          .BYTE   IPL$_SYNCH
                 082C  1636 LOCK_END:
                 082C  1637          ASSUME  <LOCK_END-LOCK_BEGIN> LE 512
```

```
                                                        .SBTTL  MOVEFU - Move data from user to system buffer
                                                ;++
                                                ;
                                                ; FUNCTIONAL DESCRIPTION:
                                                ;
                                                ;       This routine is entered as the result of a special kernel AST
                                                ;       generated by a process requesting information through $GETJPI
                                                ;       on another process.  MOVEFU is passed control information and
                                                ;       the item list in the AST packet.  Also chained into the AST
                                                ;       packet is another packet for returning the data.  This packet
                                                ;       is returned by issuing a special kernel AST to the process
                                                ;       requesting the information, to the label MOVETU in GETJPI.
                                                ;
                                                ; CALLING SEQUENCE:
                                                ;
                                                ;       JSB (as the result of a special kernel AST)
                                                ;
                                                ; INPUTS:
                                                ;
                                                ;       R0:R3 - scratch
                                                ;       R4 - PCB ADDRESS
                                                ;       R5 - AST control block address
                                                ;       Control block (see below)
                                                ;
                                                ; OUTPUTS:
                                                ;
                                                ;       None
                                                ;
                                                ; ROUTINE VALUE:
                                                ;
                                                ;       None
                                                ;
                                                ; SIDE EFFECTS:
                                                ;
                                                ;       If the process requesting information still exists, a special
                                                ;       kernel AST is issued to address MOVETU to process the filled
                                                ;       information packets.
                                                ;
                                                ;--
                                                        .enable lsb
                                                MOVEFU:
        2FF0 8F     BB  082C                            PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11,FP>
        5D   5E     D0  0830                            MOVL    SP,FP                           ; set address of local storage
     5E   D8 AE     DE  0833                            MOVAL   LOCAL_SPACE(SP),SP              ; allocate local storage
  0C A5   28 A5     D0  0837                            MOVL    ACB_L_OPID(R5),ACB$L_PID(R5) ; turn the block around
18 A5   08C1'CF     9E  083C                            MOVAB   W^MOVETU,ACB$L_KAST(R5) ; new AST routine
  08 A5   80 8F     88  0842                            BISB2   #<1@ACB$V_KAST>,ACB$B_RMOD(R5)  ; set special kernel bit again
     5A   30 A5     D0  0847                            MOVL    ACB_L_COUNT(R5),R10             ; get item count
     5B   1C A5     D0  084B                            MOVL    ACB_L_DADDR(R5),R11             ; get data block address
     57   34 A5     DE  084F                            MOVAL   ACB_L_ILIST(R5),R7              ; point to start of item list
        59   54     D0  0853                            MOVL    R4,R9                           ; setup for call to CHECKITEM
                        0856                    ;
                        0856                    ; Loop through item descriptor list, moving data to the system buffer
                        0856                    ;
        56   87     3C  0856                    10$:    MOVZWL  (R7)+,R6                        ; get user buffer size
     EC AD   57     D0  0859                            MOVL    R7,PHDTEMP(FP)                  ; save address of item identifier
        51   87     3C  085D                            MOVZWL  (R7)+,R1                        ; item identifier
```

```
                     FD21   30  0860  1696          BSBW    CHECKITEM                ; get address and size of item
                     FEC8   30  0863  1697          BSBW    CHECK_SPC                ; check for special types
                 55    02   D1  0866  1698          CMPL    #CSTRING,R5              ; counted string?
                       03   12  0869  1699          BNEQ    20$                      ; branch if not
                 53    84   9A  086B  1700          MOVZBL  (R4)+,R3                 ; get length and skip length byte
                                086E  1701
                EC BD    53   B0  086E  1702 20$:     MOVW    R3,@PHDTEMP(FP)          ; clobber item code with source size
        6B   56   00     64   53  2C  0872  1703          MOVC5   R3,(R4),#0,R6,(R11)      ; move data to system buffer
                 5B    53   D0  0878  1704          MOVL    R3,R11                   ; update system buffer pointer
                 57    08   C0  087B  1705          ADDL    #8,R7                    ; update item descriptor pointer
                                087E  1706                                           ; by skipping user buffer addresses
                   D5 5A    F5  087E  1707          SOBGTR  R10,10$                  ; decrement item count and loop
                                0881  1708 ;
                                0881  1709 ; We have moved all the data to the system buffer. Restore registers, and
                                0881  1710 ; check to see if the requesting process is still active, before we queue the
                                0881  1711 ; return kernel AST.
                                0881  1712 ;
                5E    28 AE      DE  0881  1713          MOVAL   -LOCAL_SPACE(SP),SP      ; remove local storage from stack
                   2FF0 8F      BA  0885  1714          POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11,FP>
                                0889  1715 30$:     SETIPL  50$                      ; raise IPL to synch, lock code.
                51    28 A5    3C  0890  1716          MOVZWL  ACB_L_OPID(R5),R1        ; old PID
           51   00000000'FF41    D0  0894  1717          MOVL    @SCH$GL_PCBVEC[R1],R1    ; PCB address associated.
                0C A5    60 A1    D1  089C  1718          CMPL    PCB$L_PID(R1),ACB$L_PID(R5)      ; same PID in both places?
                               11   12  08A1  1719          BNEQ    40$                      ; error if not.
                0C 24 A1   01   E0  08A3  1720          BBS     #PCB$V_DELPEN,PCB$L_STS(R1),40$ ; error if delete pending
                               52   D4  08A8  1721          CLRL    R2                       ; null priority increment
                  00000000'EF   16  08AA  1722          JSB     SCH$QAST                 ; queue the AST
                                08B0  1723          SETIPL  #IPL$_ASTDEL             ; drop back to AST delivery level
                               05   08B3  1724          RSB                              ;
                                08B4  1725 ;
                                08B4  1726 ; If the process did not exist, or was marked for delete, deallocate the
                                08B4  1727 ; blocks and return.
                                08B4  1728 ;
                                08B4  1729 40$:     SETIPL  #IPL$_ASTDEL             ; drop back to AST delivery level
                50    55   D0  08B7  1730          MOVL    R5,R0                    ; get AST block address
                                08BA  1731 DEANONPAGED:                              ; local point for all calls to EXE$D...
           00000000'GF   17  08BA  1732          JMP     G^EXE$DEANONPAGED        ; deallocate it and exit
                                08C0  1733 ;
                                08C0  1734 ; The preceding code must raise IPL to synchronize access to process database,
                                08C0  1735 ; but since it is paged it must be locked in memory. The usage of the SETIPL
                                08C0  1736 ; macro above, both raises IPL and faults the code into memory.
                                08C0  1737 ;
                                08C0  1738 50$:
                               08   08C0  1739          .BYTE   IPL$_SYNCH               ; end of locked code region
                                08C1  1740          ASSUME  <.-30$> LE 512           ; only 512 bytes can be locked
                                08C1  1741          .disable lsb
                                08C1  1742
```

```
                                                B 1
                     08C1  1744          .SBTTL  MOVETU - Move data from system buffer to user
                     08C1  1745 ;++
                     08C1  1746 ;
                     08C1  1747 ; FUNCTIONAL DESCRIPTION:
                     08C1  1748 ;
                     08C1  1749 ;       MOVETU is entered as the result of a special kernel AST queued by
                     08C1  1750 ;       the routine MOVEFU from the process we were requesting information
                     08C1  1751 ;       from on a GETJPI system service.  The data buffer has been filled,
                     08C1  1752 ;       and now we must move that data from the system buffer to the user.
                     08C1  1753 ;
                     08C1  1754 ;       Prior to storing the data, we check to see if the copy of PHD$L_IMGCNT
                     08C1  1755 ;       that was saved in the packet is the same as that in the process header.
                     08C1  1756 ;       If they are not equal, it means the image that issued the GETJPI service
                     08C1  1757 ;       has exited, and a new image is in memory; we should not move the data
                     08C1  1758 ;       to the user.
                     08C1  1759 ;
                     08C1  1760 ; CALLING SEQUENCE:
                     08C1  1761 ;
                     08C1  1762 ;       JSB (as the result of a special kernel AST)
                     08C1  1763 ;
                     08C1  1764 ; INPUTS:
                     08C1  1765 ;
                     08C1  1766 ;       R0:R3 - scratch
                     08C1  1767 ;       R4 - PCB address
                     08C1  1768 ;       R5 - AST control block address
                     08C1  1769 ;       Control block data
                     08C1  1770 ;
                     08C1  1771 ; OUTPUTS:
                     08C1  1772 ;
                     08C1  1773 ;       none
                     08C1  1774 ;
                     08C1  1775 ; ROUTINE VALUE:
                     08C1  1776 ;
                     08C1  1777 ;       none
                     08C1  1778 ;
                     08C1  1779 ; SIDE EFFECTS:
                     08C1  1780 ;
                     08C1  1781 ;       Attempts to move data to user buffers, as requested by original
                     08C1  1782 ;       GETJPI request. May cause setting of event flags, IOSB, and
                     08C1  1783 ;       possibly an AST to the requestor.  Errors in processing result
                     08C1  1784 ;       in an attempt to post the error status in the IOSB, if specified.
                     08C1  1785 ;--
                     08C1  1786 MOVETU:
                     08C1  1787 ;
                     08C1  1788 ; See if PHD$L_IMGCNT has what we think it has in it, and free the blocks
                     08C1  1789 ; and exit if it doesn't; if not equal, a different image is running!
                     08C1  1790 ;
 53   00000000'EF    D0  08C1  1791          MOVL    CTL$GL_PHD,R3                   ; get process header address
  2C A5    00F4 C3   D1  08C8  1792          CMPL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R5) ; see if the same thing.
               1A    13  08CE  1793          BEQL    10$                             ; go move data if equal
    03 0B A5    06   E1  08D0  1794          BBC     #ACB$V_QUOTA,ACB$B_RMOD(R5),5$ ; has AST quota been charged?
          38 A4    B6  08D5  1795          INCW    PCB$W_ASTCNT(R4)                ; give it back
          50    55   D0  08D8  1796 5$:     MOVL    R5,R0                           ; get address of AST block
     51   0080 C4   D0  08DB  1797          MOVL    PCB$L_JIB(R4),R1                ; get address of JIB
       52    08 A0   3C  08E0  1798          MOVZWL  ACB$W_SIZE(R0),R2               ; convert count to longword
       20 A1   52   C0  08E4  1799          ADDL    R2,JIB$L_BYTCNT(R1)             ; restore buffer quota
               D0   11  08E8  1800          BRB     DEANONPAGED                     ; deallocate AST block and exit
```

```
                                                C 1
                     08EA  1801
                     08EA  1802 10$:
    0FF0 8F    BB    08EA  1803          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>
59    0B A5    9A    08EE  1804          MOVZBL  ACB$B_RMOD(R5),R9       ; get requester's access mode
5A    30 A5    D0    08F2  1805          MOVL    ACB_L_COUNT(R5),R10     ; get item count
51    1C A5    D0    08F6  1806          MOVL    ACB_L_DADDR(R5),R1      ; get data buffer address
56    34 A5    DE    08FA  1807          MOVAL   ACB_L_ILIST(R5),R6      ; get starting address of the list
                     08FE  1808 ;
                     08FE  1809 ; Loop through item descriptor list, moving data to user buffer(s)
                     08FE  1810 ;
                     08FE  1811 20$:
    57   86    3C    08FE  1812          MOVZWL  (R6)+,R7                ; user buffer length
    58   86    3C    0901  1813          MOVZWL  (R6)+,R8                ; actual data length
    55   86    D0    0904  1814          MOVL    (R6)+,R5                ; user buffer address
                     0907  1815 ;
                     0907  1816 ; Check that requester still has write access to his buffer
                     0907  1817 ;
    50   55    D0    0907  1818          MOVL    R5,R0                   ; buffer address to R0
               51    DD    090A  1819          PUSHL   R1                      ; Save R1
    51   57    D0    090C  1820          MOVL    R7,R1                   ; and size to R1
    53   59    D0    090F  1821          MOVL    R9,R3                   ; use access mode value from ACB for PROBE
00000000'EF      16    0912  1822          JSB     EXE$PROBEW              ; check write accessibility of buffer
               51 8ED0    0918  1823          POPL    R1                      ; Restore R1
      22 50    E9    091B  1824          BLBC    R0,50$                  ; get out if buffer inaccessible
                     091E  1825 ;
                     091E  1826 ; Now actually move the data
                     091E  1827 ;
65  61   57    28    091E  1828          MOVC3   R7,(R1),(R5)            ; move data to user buffer
    50   86    D0    0922  1829          MOVL    (R6)+,R0                ; get address to store actual length
         11    13    0925  1830          BEQL    40$                     ; branch if no length wanted
                     0927  1831          IFNOWRT #2,(R0),50$,R9          ; requester still have access to buffer?
    57   58    D1    092D  1832          CMPL    R8,R7                   ; actual data length less than user's?
         03    15    0930  1833          BLEQ    30$                     ; branch if yes - use actual length
    58   57    D0    0932  1834          MOVL    R7,R8                   ; use user buffer length
    60   58    B0    0935  1835 30$:     MOVW    R8,(R0)                 ; return buffer length
      C3 5A    F5    0938  1836 40$:     SOBGTR  R10,20$                 ; decrement item count and loop
    50   01    3C    093B  1837          MOVZWL  #SS$_NORMAL,R0          ; set successful completion
         03    11    093E  1838          BRB     60$                     ;
    50   0C    3C    0940  1839 50$:     MOVZWL  #SS$_ACCVIO,R0          ; set access violation failure
                     0943  1840 ;
                     0943  1841 ; Restore original registers, set the event flag, and post completion status
                     0943  1842 ;
    0FF0 8F    BA    0943  1843 60$:     POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11>
               50    DD    0947  1844          PUSHL   R0                      ; save status
 53   20 A5    D0    0949  1845          MOVL    ACB_L_EFN(R5),R3        ; get event flag number
 51   60 A4    D0    094D  1846          MOVL    PCB$L_PID(R4),R1        ; and PID for process
               52    D4    0951  1847          CLRL    R2                      ; set null priority increment
00000000'GF      16    0953  1848          JSB     G^SCH$POSTEF            ; set the event flag
               50 8ED0    0959  1849          POPL    R0                      ; restore exit status
 53   24 A5    D0    095C  1850          MOVL    ACB_L_IOSB(R5),R3       ; possible IOSB address?
         0A    13    0960  1851          BEQL    70$                     ; branch if none supplied
                     0962  1852          IFNOWRT #4,(R3),70$,ACB$B_RMOD(R5) ; check if IOSB still accessable
    63   50    D0    0969  1853          MOVL    R0,(R3)                 ; store completion status
                     096C  1854 ; Return the BYTCNT quota to the caller
                     096C  1855 ;
  52   08 A5    3C    096C  1856 70$:     MOVZWL  ACB$W_SIZE(R5),R2       ; convert to longword
51  0080 C4    D0    0970  1857          MOVL    PCB$L_JIB(R4),R1        ; get JIB address
```

```
                                                     D 1
                     20 A1   52   C0  0975  1858          ADDL    R2,JIB$L_BYTCNT(R1)     ; restore buffer quota
                                      0979  1859 ;
                                      0979  1860 ; If an AST was specified, queue it to caller and return.
                                      0979  1861 ;
                        10 A5   D5    0979  1862          TSTL    ACB$L_AST(R5)           ; is an address supplied?
                           08   13    097C  1863          BEQL    80$                     ; branch if not.
                           52   D4    097E  1864          CLRL    R2                      ; no priority increment
                 00000000'EF   17    0980  1865          JMP     SCH$QAST                ; queue AST to user and exit
                                      0986  1866 ;
                                      0986  1867 ; No AST specified, deallocate the AST control block and return.
                                      0986  1868 ;
                           50   55    D0    0986  1869 80$:     MOVL    R5,R0                   ; set the address of the AST block
                              FF2E   31    0989  1870          BRW     DEANONPAGED             ; deallocate the block and exit
```

```
                                                                E 1
                     098C  1872          .SBTTL  NAMPID - Get specified process ID
                     098C  1873 ;++
                     098C  1874 ;
                     098C  1875 ; FUNCTIONAL DESCRIPTION:
                     098C  1876 ;
                     098C  1877 ;       Routine to convert a process name to a PID and check privileges. If a
                     098C  1878 ;       valid PID or process name is specified, the standard conversion
                     098C  1879 ;       routine EXE$NAMPID is simply called.  If, however, a PID that implies
                     098C  1880 ;       a "wildcard" PID (-1) is specified, then the next active process is
                     098C  1881 ;       chosen as the process ID to pass to EXE$NAMPID.  EXE$NAMPID then
                     098C  1882 ;       checks the requestor's privilege to obtain information about the
                     098C  1883 ;       process and returns the process's PCB address.
                     098C  1884 ;
                     098C  1885 ; INPUTS:
                     098C  1886 ;
                     098C  1887 ;       R4 = current process PCB address
                     098C  1888 ;       PIDADR(AP) = address of specified PID
                     098C  1889 ;       PRCNAM(AP) = address of specified process name descriptor
                     098C  1890 ;
                     098C  1891 ; OUTPUTS:
                     098C  1892 ;
                     098C  1893 ;       R0 = success/failure of operation
                     098C  1894 ;       R4 = current process PCB address
                     098C  1895 ;       R9 = specified process PCB address
                     098C  1896 ;       R11 = specified process PID
                     098C  1897 ;       @PIDADR(AP) = specified process PID or special "wildcard" context PID
                     098C  1898 ;--
                     098C  1899 NAMPID:
                     098C  1900          .ENABL  LSB
        56   08 AC   D0   098C  1901          MOVL    PIDADR(AP),R6           ; get PID address
             43   13   0990  1902          BEQL    20$                     ; if eql - none
                  0992  1903          IFNOWRT #4,(R6),50$             ; check access to PID
           50   66   D0   0998  1904          MOVL    (R6),R0                 ; get PID
             38   18   0998  1905          BGEQ    20$                     ; if geq - standard extended PID
                  099D  1906 ;
                  099D  1907 ; "Wildcard" type PID specified
                  099D  1908 ;
             55   50   32   099D  1909          CVTWL   R0,R5                   ; get PIX (Process Index) from PID
     50   00000000'GF   DE   09A0  1910          MOVAL   G^SCH$GL_NULLPCB,R0     ; special case handling for NULL proc
          FC AD   01   C8   09A7  1911          BISL2   #<1@JPI_V_WILD>,FLAGS(FP) ; mark wildcarding in progress
             55   B6   09AB  1912 10$:     INCW    R5                      ; increment PIX
   00000000'EF   55   B1   09AD  1913          CMPW    R5,SCH$GL_MAXPIX        ; is PIX in valid range?
             56   1A   09B4  1914          BGTRU   60$                     ; if gtru, no - no more processes
          55   55   3C   09B6  1915          MOVZWL  R5,R5                   ; clean out top half of R5
             11   13   09B9  1916          BEQL    15$                     ; if eql, this indicates actual NULL proc
 50   00000000'FF45   D0   09BB  1917          MOVL    @SCH$GL_PCBVEC[R5],R0   ; get PCB address
   00000000'8F   50   D1   09C3  1918          CMPL    R0,#SCH$GL_NULLPCB      ; unused process slot?
             DF   13   09CA  1919          BEQL    10$                     ; if eql, yes - try next one
      DA 24 A0   01   E0   09CC  1920 15$:     BBS     #PCB$V_DELPEN,PCB$L_STS(R0),10$ ; also get next one if this
                  09D1  1921                                              ; one is going away
        66   64 A0   D0   09D1  1922          MOVL    PCB$L_EPID(R0),(R6)     ; store extended PID in argument list
                  09D5  1923 ;
                  09D5  1924 ; Convert process name to PID, if specified, and check privileges
                  09D5  1925 ;
           5C   04   CO   09D5  1926 20$:     ADDL    #4,AP                   ; make PIDADR top argument
     00000000'EF   16   09D8  1927          JSB     25$                     ; get into nonpaged code
                  09DE  1928          .SAVE_PSECT                     ; save current .PSECT context
```

```
                                                09DE  1929
                                                09DE  1930 ;  The reason for jumping to the nonpaged exec rather than dynamically
                                                09DE  1931 ;  locking down pageable pages is that EXE$NAMPID cannot be entered
                                                09DE  1932 ;  above IPL 2 and the dynamic locking would cause that to happen. The
                                                09DE  1933 ;  reason that EXE$NAMPID must be entered at IPL 2 or lower is that it
                                                09DE  1934 ;  touches the caller's argument list (which contains arguments that
                                                09DE  1935 ;  could fault) and page faults are not allowed above IPL 2.
                                                09DE  1936
                                    00000000    1937          .PSECT  AEXENONPAGED                    ; EXE$NAMPID returns at IPL$_SYNCH
                                        FFFD'   30  0000    1938 25$:    BSBW    EXE$NAMPID                 ; get PCB address and check privileges
                                                    0003    1939         SETIPL  #0                         ; restore IPL - PCB is no longer locked
                                                05  0006    1940         RSB                                ; go back to paged code
                                                    0007    1941
                                                000009DE    1942         .RESTORE_PSECT                     ; get paged .PSECT context back
                                              5C    04  C2  09DE    1943         SUBL    #4,AP              ; restore argument pointer
                                              5B    51  D0  09E1    1944         MOVL    R1,R11             ; save PID
                                        07    FC    AD  E1  09E4    1945         BBC     #JPI_V_WILD,FLAGS(FP),30$ ; "wildcard" PID specified?
                                              66    51  B0  09E9    1946         MOVW    R1,(R6)            ; restore process index context
                                           02    A6   01  AE  09EC    1947         MNEGW   #1,2(R6)           ; else, set continuation context
                                                      09F0  1948 ;
                                                      09F0  1949 ; Check PID address and return
                                                      09F0  1950 ;
                                                 0C   50  E9  09F0    1951 30$:    BLBC    R0,40$             ; branch if error
                                                59    54  D0  09F3    1952         MOVL    R4,R9              ; save PCB address
                                                00'   5B  B1  09F6    1953         CMPW    R11,S^#SCH$C_SWPPIX ; is PID a normal one?
                                                      04  1A  09F9    1954         BGTRU   40$                ; if GTRU, yes
                                              FC    AD    02  C8  09FB    1955         BISL2   #<1@JPI_V_NULLSWAP>,FLAGS(FP) ; indicate Null or Swapper process
                                        54    00000000'EF   D0  09FF    1956 40$:    MOVL    SCH$GL_CURPCB,R4   ; restore current PCB address
                                                      05  0A06    1957         RSB
                                                      0A07    1958
                                                   50 0C  3C  0A07    1959 50$:    MOVZWL  #SS$_ACCVIO,R0     ; set access violation
                                                      F3  11  0A0A    1960         BRB     40$                ;
                                              50    09A8 8F  3C  0A0C    1961 60$:    MOVZWL  #SS$_NOMOREPROC,R0 ; set no more processes
                                                      EC  11  0A11    1962         BRB     40$                ;
                                                          0A13    1963
                                                          0A13    1964         .END
```

```
$$T1                    = 00000000
$XX$                    = 00000000
ACB$B_RMOD              = 0000000B
ACB$B_TYPE              = 0000000A
ACB$L_AST               = 00000010
ACB$L_ASTPRM            = 00000014
ACB$L_KAST              = 00000018
ACB$L_PID               = 0000000C
ACB$V_KAST              = 00000007
ACB$V_QUOTA             = 00000006
ACB$W_SIZE              = 00000008
ACB_C_IDESC             = 0000000C
ACB_L_COUNT               00000030
ACB_L_DADDR               0000001C
ACB_L_EFN                 00000020
ACB_L_ILIST               00000034
ACB_L_IMGCNT              0000002C
ACB_L_IOSB                00000024
ACB_L_OPID                00000028
ADRTBL                    00000006 R      02
ASTADR                  = 00000018
ASTPRM                  = 0000001C
BITDEF                  = FFFFFFE8
BOOLE                   = 00000003
BSTRING                 = 00000001
CHECKITEM                 00000584 R      02
CHECK_SPC                 0000072E R      02
CSTRING                 = 00000002
CTL$AL_CMCNTX             ******** X    02
CTL$AL_FINALEXC           ******** X    02
CTL$AQ_EXCVEC             ******** X    02
CTL$GB_MSGMASK            ******** X    02
CTL$GL_CMHANDLR           ******** X    02
CTL$GL_CREPRC_FLAGS       ******** X    02
CTL$GL_IMGHDRBF           ******** X    02
CTL$GL_PHD                ******** X    02
CTL$GL_SITESPEC           ******** X    02
CTL$GL_UAF_FLAGS          ******** X    02
CTL$GL_VIRTPEAK           ******** X    02
CTL$GL_VOLUMES            ******** X    02
CTL$GL_WSPEAK             ******** X    02
CTL$GQ_LOGIN              ******** X    02
CTL$GQ_PROCPRIV           ******** X    02
CTL$GT_CLINAME            ******** X    02
CTL$GT_TABLENAME          ******** X    02
CTLTBL                    00000010 R      02
DEANONPAGED               000008BA R      02
DIRCNT                  = FFFFFFF8
DSTRING                 = 00000004
DYN$C_ACB               = 00000002
EFN                     = 00000004
EXE$ALONONPAGED           ********   X    02
EXE$BUFFRQUOTA            ********   X    02
EXE$CHKIMAGNAME           000007BB RG     02
EXE$DEANONPAGED           ********     X  02
EXE$GETJPI                00000000 RG     03
EXE$IPID_TO_EPID          ********   X    02
EXE$NAMPID                ********   X    04
EXE$PROBEW                ********   X    02
EXE_GETJPI                00000219 R      02
EXTFLD                    0000066E R      02
FLAGS                   = FFFFFFFC
GRET                      0000033E R      02
IFD$Q_CURPROG           = 00000014
IOSB                    = 00000014
IPL$_ASTDEL             = 00000002
IPL$_SYNCH              = 00000008
ITMLST                  = 00000010
JIB$B_JOBTYPE           = 00000068
JIB$L_BYTCNT            = 00000020
JIB$L_BYTLM             = 00000024
JIB$L_MPID              = 00000054
JIB$L_PGFLCNT           = 0000003C
JIB$L_PGFLQUOTA         = 00000038
JIB$T_ACCOUNT           = 00000018
JIB$T_USERNAME          = 0000000C
JIB$W_ENQCNT            = 0000004C
JIB$W_ENQLM             = 0000004E
JIB$W_FILCNT            = 00000030
JIB$W_FILLM             = 00000032
JIB$W_MAXDETACH         = 00000052
JIB$W_MAXJOBS           = 00000050
JIB$W_PRCCNT            = 00000044
JIB$W_PRCLIM            = 00000046
JIB$W_SHRFLIM           = 0000004A
JIB$W_TQCNT             = 00000034
JIB$W_TQLM              = 00000036
JPI$C_ADRTYPE           = 00000001
JPI$C_CTLTYPE           = 00000002
JPI$C_JIBTYPE           = 00000007
JPI$C_MAXSTRUC          = 00000006
JPI$C_PCBFLDTYPE        = 00000005
JPI$C_PCBTYPE           = 00000003
JPI$C_PHDFLDTYPE        = 00000006
JPI$C_PHDTYPE           = 00000004
JPI$K_BATCH             = 00000002
JPI$K_INTERACTIVE       = 00000003
JPI$K_NETWORK           = 00000001
JPI$K_OTHER             = 00000000
JPI$_ACCOUNT            = 00000203
JPI$_APTCNT             = 0000030A
JPI$_ASTACT             = 00000300
JPI$_ASTCNT             = 0000030E
JPI$_ASTEN              = 00000301
JPI$_ASTLM              = 00000409
JPI$_AUTHPRI            = 00000418
JPI$_AUTHPRIV           = 00000412
JPI$_BIOCNT             = 0000030F
JPI$_BIOLM              = 00000310
JPI$_BUFIO              = 0000040C
JPI$_BYTCNT             = 00000311
JPI$_BYTLM              = 0000031A
JPI$_CHAIN              = FFFFFFFF
JPI$_CLINAME            = 0000020A
```

```
JPI$_CPULIM             =  0000040D
JPI$_CPUTIM             =  00000407
JPI$_CREPRC_FLAGS       =  0000020C
JPI$_CURPRIV            =  00000400
JPI$_DFPFC              =  00000406
JPI$_DFWSCNT            =  00000403
JPI$_DIOCNT             =  00000312
JPI$_DIOLM              =  00000313
JPI$_DIRIO              =  0000040B
JPI$_EFCS               =  00000317
JPI$_EFCU               =  00000318
JPI$_EFWM               =  00000316
JPI$_ENQCNT             =  0000031F
JPI$_ENQLM              =  00000320
JPI$_EXCVEC             =  00000100
JPI$_FILCNT             =  00000314
JPI$_FILLM              =  0000040F
JPI$_FINALEXC           =  00000101
JPI$_FREPOVA            =  00000404
JPI$_FREP1VA            =  00000405
JPI$_FREPTECNT          =  00000415
JPI$_GPGCNT             =  0000030C
JPI$_GRP                =  00000308
JPI$_IMAGECOUNT         =  0000041A
JPI$_IMAGNAME           =  00000207
JPI$_IMAGPRIV           =  00000413
JPI$_JOBPRCCNT          =  0000031E
JPI$_JOBTYPE            =  00000323
JPI$_LASTADR            =  00000102
JPI$_LASTCTL            =  00000211
JPI$_LASTPCB            =  00000326
JPI$_LASTPCBFLD         =  00000500
JPI$_LASTPHD            =  0000041C
JPI$_LASTPHDFLD         =  00000600
JPI$_LOGINTIM           =  00000206
JPI$_MASTER_PID         =  00000325
JPI$_MAXDETACH          =  0000020E
JPI$_MAXJOBS            =  0000020F
JPI$_MEM                =  00000307
JPI$_MODE               =  00000322
JPI$_MSGMASK            =  00000209
JPI$_OWNER              =  00000303
JPI$_PAGEFLTS           =  0000040A
JPI$_PAGFILCNT          =  00000414
JPI$_PAGFILLOC          =  00000419
JPI$_PGFLQUOTA          =  0000040E
JPI$_PHDFLAGS           =  0000041B
JPI$_PID                =  00000319
JPI$_PPGCNT             =  0000030D
JPI$_PRCCNT             =  0000031B
JPI$_PRCLM              =  00000408
JPI$_PRCNAM             =  0000031C
JPI$_PRI                =  00000302
JPI$_PRIB               =  00000309
JPI$_PROCPRIV           =  00000204
JPI$_PROC_INDEX         =  00000324
JPI$_SHRFILLM           =  00000210
JPI$_SITESPEC           =  00000208
JPI$_STATE              =  00000306
JPI$_STS                =  00000305
JPI$_SWPFILLOC          =  00000321
JPI$_TABLENAME          =  0000020B
JPI$_TERMINAL           =  0000031D
JPI$_TMBU               =  0000030B
JPI$_TQCNT              =  00000315
JPI$_TQLM               =  00000410
JPI$_UAF_FLAGS          =  0000020D
JPI$_UIC                =  00000304
JPI$_USERNAME           =  00000202
JPI$_VIRTPEAK           =  00000200
JPI$_VOLUMES            =  00000205
JPI$_WSAUTH             =  00000401
JPI$_WSAUTHEXT          =  00000417
JPI$_WSEXTENT           =  00000416
JPI$_WSPEAK             =  00000201
JPI$_WSQUOTA            =  00000402
JPI$_WSSIZE             =  00000411
JPI_BIT                 =  00000003
JPI_S_NULLSWAP          =  00000001
JPI_S_STRDSC            =  00000001
JPI_S_WILD              =  00000001
JPI_V_NULLSWAP          =  00000001
JPI_V_STRDSC            =  00000002
JPI_V_WILD              =  00000000
LOCAL_SPACE             =  FFFFFFD8
LOCK_BEGIN              =  000006D3 R     02
LOCK_END                =  0000082C R     02
LOCK_IPL                =  0000082B R     02
MAXCOUNT                =  00000000 R     02
MAX_ADR_ITEM            =  00000001
MAX_CTL_ITEM            =  00000010
MAX_PCBFLD_ITEM         =  FFFFFFFF
MAX_PCB_ITEM            =  00000025
MAX_PHDFLD_ITEM         =  FFFFFFFF
MAX_PHD_ITEM            =  0000001B
MOVEFU                  =  0000082C R     02
MOVEIT                  =  0000068B R     02
MOVEPHD                 =  000006D0 R     02
MOVETU                  =  000008C1 R     02
NAMPID                  =  0000098C R     02
PCB$B_ASTACT            =  0000000C
PCB$B_ASTEN             =  0000000D
PCB$B_PRI               =  0000000B
PCB$B_PRIB              =  0000002F
PCB$L_EFCS              =  00000050
PCB$L_EFCU              =  00000054
PCB$L_EFWM              =  0000004C
PCB$L_EOWNER            =  00000068
PCB$L_EPID              =  00000064
PCB$L_JIB               =  00000080
PCB$L_PHD               =  0000006C
PCB$L_PID               =  00000060
PCB$L_STS               =  00000024
PCB$L_UIC               =  000000BC
```

```
PCB$L_WSSWP                     = 00000020
PCB$T_LNAME                     = 00000070
PCB$T_TERMINAL                  = 00000044
PCB$V_BATCH                     = 0000000E
PCB$V_DELPEN                    = 00000001
PCB$V_INTER                     = 00000019
PCB$V_NETWRK                    = 00000015
PCB$V_PHDRES                    = 00000012
PCB$V_SSRWAIT                   = 0000000A
PCB$V_SUSPEN                    = 0000000B
PCB$W_APTCNT                    = 00000030
PCB$W_ASTCNT                    = 00000038
PCB$W_BIOCNT                    = 0000003A
PCB$W_BIOLM                     = 0000003C
PCB$W_DIOCNT                    = 0000003E
PCB$W_DIOLM                     = 00000040
PCB$W_GPGCNT                    = 00000034
PCB$W_GRP                       = 000000BE
PCB$W_MEM                       = 000000BC
PCB$W_PPGCNT                    = 00000036
PCB$W_PRCCNT                    = 00000042
PCB$W_STATE                     = 0000002C
PCB$W_TMBU                      = 00000032
PCBFLDTBL                         000001D1 R      02
PCBTBL                            00000087 R      02
PHD$B_AUTHPRI                   = 0000010C
PHD$B_DFPFC                     = 00000034
PHD$L_BIOCNT                    = 00000058
PHD$L_CPULIM                    = 0000005C
PHD$L_CPUTIM                    = 00000038
PHD$L_DIOCNT                    = 00000054
PHD$L_FREPOVA                   = 00000028
PHD$L_FREP1VA                   = 00000030
PHD$L_FREPTECNT                 = 0000002C
PHD$L_IMGCNT                    = 000000F4
PHD$L_PAGEFLTS                  = 0000004C
PHD$L_PAGFIL                    = 0000001C
PHD$Q_AUTHPRIV                  = 000000E0
PHD$Q_IMAGPRIV                  = 000000E8
PHD$Q_PRIVMSK                   = 00000000
PHD$W_ASTLM                     = 00000040
PHD$W_DFWSCNT                   = 0000001A
PHD$W_FLAGS                     = 00000036
PHD$W_WSAUTH                    = 0000000A
PHD$W_WSAUTHEXT                 = 00000014
PHD$W_WSEXTENT                  = 00000016
PHD$W_WSLIST                    = 00000008
PHD$W_WSQUOTA                   = 00000018
PHD$W_WSSIZE                    = 00000050
PHDFLDTBL                         000001D1 R      02
PHDTBL                            00000145 R      02
PHDTEMP                         = FFFFFFEC
PIDADR                          = 00000008
PR$_IPL                         = 00000012
PRCNAM                          = 0000000C
PRI$_TICOM                      = 00000004
PSL$C_USER                      = 00000003
PSL$M_IPL                       = 001F0000
PSL$S_PRVMOD                    = 00000002
PSL$V_PRVMOD                    = 00000016
RESCAN                            000003AC R      02
RSN$_NPDYNMEM                   = 00000003
SCH$CHSEP                         ********    X   02
SCH$CLREF                         ********    X   02
SCH$C_COM                       = 0000000C
SCH$C_COMO                      = 0000000D
SCH$C_MWAIT                     = 00000002
SCH$C_SUSP                      = 00000009
SCH$C_SUSPO                     = 0000000A
SCH$C_SWPPIX                      ********    X   02
SCH$GL_CURPCB                     ********    X   02
SCH$GL_MAXPIX                     ********    X   02
SCH$GL_NULLPCB                    ********    X   02
SCH$GL_PCBVEC                     ********    X   02
SCH$POSTEF                        ********    X   02
SCH$QAST                          ********    X   02
SCH$RWAIT                         ********    X   02
SCRATCH                         = FFFFFFD8
SGN$GW_MAXPRCCT                   ********    X   02
SPC_IMAGNAME                      000007DC R      02
SPC_MASTER_PID                    000007AB R      02
SPC_MODE                          00000758 R      02
SPC_PRI                           00000750 R      02
SPC_PROC_INDEX                    0000079A R      02
SPC_WORKSET                       00000778 R      02
SPECIAL                           000001D1 R      02
SPECIAL_LEN                     = 0000000C
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_EXQUOTA                     = 0000001C
SS$_INSFMEM                     = 00000124
SS$_NOMOREPROC                  = 000009A8
SS$_NONEXPR                     = 000008E8
SS$_NORMAL                      = 00000001
SS$_SUSPENDED                   = 000003A4
STEP                            = 00000005
SYS$DCLAST                        ********  GX    02
VALUE                           = 00000000
WSLIST                          = FFFFFFF4
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000034 (52.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YF$$SYSGETJPI	00000A13 (2579.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
YEXEPAGED	00000005 (5.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
AEXENONPAGED	00000007 (7.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:01.20
Command processing	121	00:00:00.68	00:00:04.14
Pass 1	518	00:00:27.14	00:01:02.51
Symbol table sort	0	00:00:02.07	00:00:07.15
Pass 2	343	00:00:06.17	00:00:20.11
Symbol table output	38	00:00:00.25	00:00:01.44
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1054	00:00:36.43	00:01:36.59

The working set limit was 2100 pages.
135941 bytes (266 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1273 non-local and 119 local symbols.
1964 source lines were read in Pass 1, producing 28 object records in Pass 2.
51 pages of virtual memory were used to define 36 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]SYSBLDMLB.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	18
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	30

1632 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSGETJPI/OBJ=OBJ$:SYSGETJPI MSRC$:SYSGETJPI/UPDATE=(ENH$:SYSGETJPI)+EXECML$/LIB+SYS$LIBRARY:SYSBLDMLB/LIB

SYSGETJPI
LIS

SYSERAPAT
LIS

SYSFAO
LIS

SYSGETDVI
LIS

SYSEXIT
LIS

SYSEVTSRV
LIS

SYSFORCEX
LIS

SYSGETSYI
LIS

SYSGETPTI
LIS

SYSGETTIM
LIS

SYSIMGFIX
LIS

SYSGETLKI
LIS

SYSGETMSG
LIS

SYSIMGACT
LIS